Technologie sieciowe 5

Gabriel Wechta

12.06.2020

1 Zadanie

W skrypcie należy zmienić "lukim" na "localhost" aby zadziałał.
Ponadto należy dodać plik index.html.
Dodałem również linijkę informującą o wysłaniu.

2 Zadanie

Po wpisaniu w pole URL przeglądarki adresu wygenerowanego przez skrypt, otrzymuję treść index.html.

Do zadania 3 i 4 użyję webowego frameworka pythona - falcon.

Adres: localhost:8000.

Jeżeli chce Pan go przetestować, polecam zainstalować:

```
$ pip install falcon
$ pip install gunicorn
```

3 Zadanie

Program: responder.py.

```
$ gunicorn responder:api
```

Niestety obiektowość falcona powoduję, że aplikacja musi mieć przeciążoną metodę na każdy header HTTP: Falcon obsługuje headery z RFC 7231 i RFC 5789, to jest: GET, HEAD, POST, PUT, DELETE, CONNECT, OPTIONS, TRACE, i PATCH.
Dla powyższych aplikacja ustawia pole req.body na string zawierający header i ją wysyła.

4 Zadanie

Program: prim.py.

```
$ gunicorn prim:api
```

Mamy trzy strony:

1. 127.0.0.1:8000
2. 127.0.0.1:8000/images
3. 127.0.0.1:8000/talker

Images wyświetla obraz zapisany w katalogu, poprzez wczytanie images.html. Zaś Talker reaguje na różne headery.
Niestety niektóre adresy ścieżek muszą być lokalne. W celu poprawienia tego, należy pozmieniać ścieżki.

5 Zadanie

Do przechwytywania komunikatów, użyłem WireSharka na Loopback: lo. Zauważyłem, że podczas standardowego zapytania z przeglądarki generowane jest GET HTTP 1.1, parę pakietów TCP i ponownie HTTP, tym razem HTTP 1.1 200 OK w przypadku sukcesu.
WireShark pozwala przeczytać słowo po słowie treść protokołu HTTP, w którym wysyłany jest plik html.
W przypadku wyświetlania obrazu WireShark pokazuje większy ruch. Najpierw wyświetla stronę (według html) potem przechodzi do kolejnego żądania o obraz.
Jeżeli chodzi o Talkera, wysłanie headera zaimplementowanego w prim.py zwraca spodziewaną odpowiedź, niezaimplementowanego error 405, czego można było się spodziewać.

Screen z analizy dołączam do sprawozdania – screen.png.