Computação Distribuída
2018 / 2019

Licenciatura em Engenharia Informática

Mini-projecto #2 – Servidor de HTTP/1.1

Neste mini-projecto pretende-se construir um servidor HTTP usando *sockets* TCP/IP. Este servidor irá implementar algumas das funcionalidades mais relevantes do protocolo HTTP/1.1.

Funcionamento geral

Após iniciar, o servidor deverá aguardar pela ligação dos clientes. Estes clientes serão essencialmente *browsers*, mas poderão ser quaisquer outras aplicações que reconheçam o protocolo HTTP/1.1.

É importante garantir que o servidor é capaz de lidar com múltiplas conexões em paralelo e que respeita a semântica de conexões HTTP/1.1, nomeadamente manter o *socket* aberto após o envio de cada resposta.

Protocolo e formato de mensagens

O servidor deverá comunicar com os clientes usando o protocolo HTTP. Para mais detalhes, aconselha-se a leitura dos seguintes documentos:

- HTTP Made Really Easy: http://www.jmarshall.com/easy/http/
- Especificação do protocolo HTTP/1.1: https://www.w3.org/Protocols/rfc2616/rfc2616.html

Funcionalidades

Segue uma lista de requisitos que deverão ser implementados. Aconselha-se a atenção aos detalhes, como por exemplo, a utilização dos *status codes* apropriados, etc:

1. **Obtenção de recursos**: Deverá ser possível ao servidor retornar páginas *html* e imagens em formato *jpg* e *png* que deverão estar localizados dentro de uma pasta de nome *htdocs*.. Irá ser testado a obtenção de todos os recursos no ficheiro *htdocs.zip* disponível no moodle.

2. **Status codes apropriados:** O servidor deverá suportar e retornar correctamente os códigos de status “200 Ok”, “400 Bad Request”, “403 Forbidden” e “404 Not Found”. Assuma que todos os ficheiros dentro da pasta “/private/” são privados e deverão retornar o 403.

3. **Suporte a verbos HTTP:** Para além dos métodos GET e HEAD, o servidor deverá também lidar com o método POST. Neste caso, o browser deverá ser capaz de receber dados formatados em "application/x-www-form-urlencoded" e deverá, como resposta retornar os dados em *json*.

   Usando o formulário em *htdocs/public/form.html* (disponível no moodle), o servidor deverá retornar, por exemplo, o seguinte json: *{"firstname": "Hello", "lastname": "World"}*.

4. **Cabeçalhos HTTP:** O servidor deverá reconhecer e utilizar, onde fizer sentido, os cabeçalhos HTTP mais relevantes, nomeadamente, *Content-Length*, *Content-Type* e *Date*.

5. **Fecho de ligações:** O servidor deverá fechar ligações quando surgir o header "*Connection: close*" (ver protocolo HTTP/1.1). Deverá também fechar as ligações que estiverem inactivas durante mais de 10 segundos.

6. **Suporte a caches:** Os *n* recursos mais requisitados deverão ser mantidos em cache (use *n=2*). Para recursos fora da cache faça um *sleep* de 100 ms de modo a simular um atraso devido à leitura no disco.

7. **Log:** Por cada pedido efectuado deverá guardar o endereço IP e o *url* num ficheiro com o nome "log.txt".

Entrega e avaliação

Todos os ficheiros deverão ser colocados num **ficheiro zip** (com o número de todos os elementos do grupo) e submetido via *moodle* **até às 23:55 do dia 19/Maio/2019**.

Deverá também ser colocado no zip um **ficheiro pdf** com a identificação dos alunos, extras implementados, a descrição geral da solução incluindo diagramas que possa considerar relevantes, e a identificação de compromissos que tiverem que tomar. Este documento deverá ser mantido curto e directo (1-2 páginas).

Irá considerar-se a seguinte grelha de avaliação:

Correcção da solução	10 val.
Qualidade e modularização do código	03 val.
Documentação	02 val.
Extras (~1 valor por extra)	05 val.

Alguns extras a considerar: (1) outros formatos de ficheiros de imagem, (2) ficheiros de áudio, (3) streaming de vídeo, (4) autenticação para aceder à pasta "private", (5) utilização de outros *headers* relevantes, (6) outros métodos HTTP relevantes, (7) programação assíncrona no lado do servidor, (8) etc.., sejam criativos!

Bom trabalho!