



Licenciatura em Engenharia Informática e de Computadores

Redes de Computadores

Relatório da 1ª Fase

Abril 2023

Trabalho realizado por:

A43842 André Monteiro

A50562 Umera Aktar

A50452 Sara Pereira

Turma: LEIC24D

Docente: Professor Luís Mata

Objetivo:

Neste trabalho tivemos como objetivo aplicar o que aprendemos nas aulas da cadeira de Redes de Computadores.

Durante a realização deste trabalho criamos um Web Client e um Web Server e estabelecemos uma ligação entre eles.

Desenvolvimento do trabalho:

1. Criação de um Web Server através do XAMPP

Através da aplicação XAMPP foi possível criar o Web Server necessário para fazer a ligação. Nas imagens seguintes é possível ver o sucesso da criação do servidor:

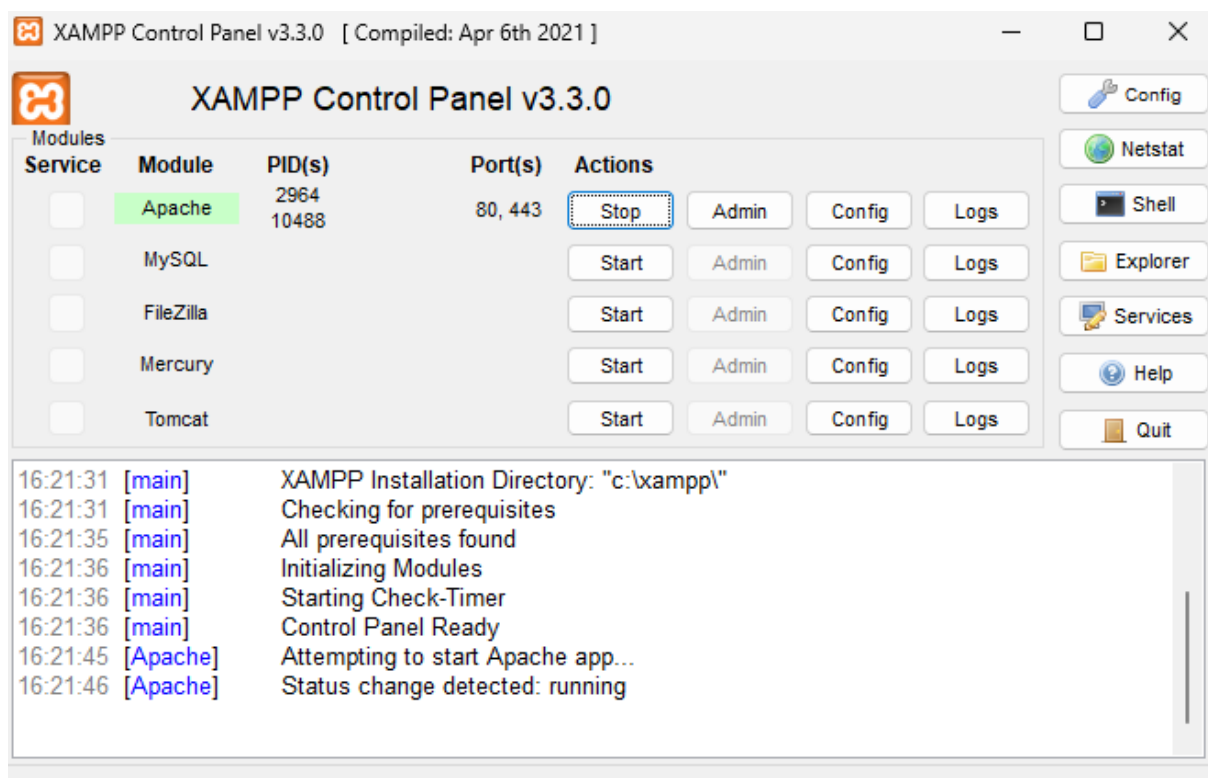


Figura 1

Tal como referido no enunciado, foi colocado o endereço <http://127.0.0.1>, demonstrando que o Web Server foi implementado com sucesso.

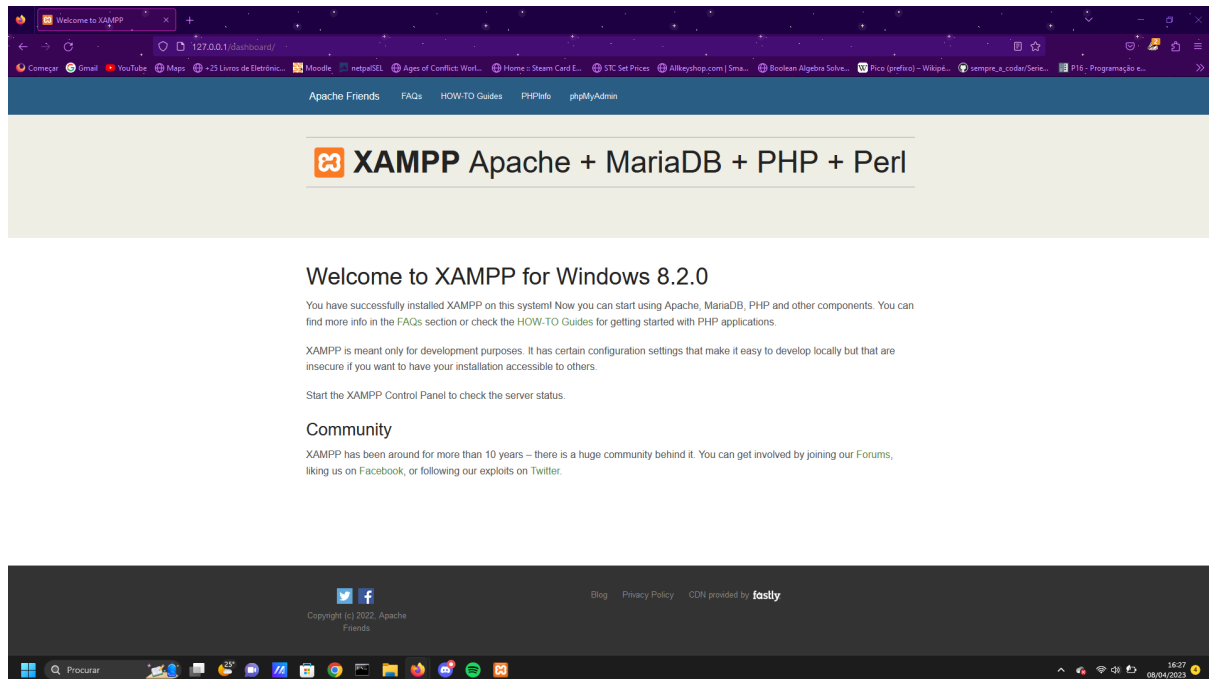


Figura 2

2. Criação de um Web Client

O passo seguinte do nosso trabalho foi desenvolver um Web Client. Decidimos fazê-lo na linguagem Kotlin junto com as bibliotecas Socket e Scanner.

- Código:

```
import java.net.Socket
import java.util.*

fun main() {
    val port = 80
    val url = "www.isel.pt"
    val client = Socket(url, port)
    val http = "HTTP/1.0"
    makeRequest(client, url, http)
    client.close()
}

fun makeRequest(client: Socket, link: String, http: String) {
    var url = link
    client.outputStream.write("GET / $http\r\nHost: $url\r\n\r\n".toByteArray())
    //faz o request do get com a versão HTTP especificada e para o link especificado
    var response = emptyArray<String>()
    val scanner = Scanner(client.getInputStream())

    // recolhe e imprime a resposta obtida
    while (scanner.hasNextLine()) {
        response += scanner.nextLine()
        if (response.last() == "") break
        println(response.last())
    }

    //no caso segundo request o nosso código não reconhece a resposta, e se não o
    paramos dá erro
    if (response.isEmpty()) return
}
```

```
// verifica qual o valor do primeiro dígito do código de resposta e faz uma ação
dependente deste valor
when (response[0].split(" ")[1][0]) {
  '1', '2' -> return
  '3' -> {                                     //vai buscar o novo link e faz um novo request
    for (i in response) {
      val s = i.split(" ")
      if (s[0] == "Location:") {
        url = s[1]
        break
      }
    }
    makeRequest(client, url, http)
  }

  '4' -> println("Try again with a different request.")
  '5' -> {
    //se o último dígito for maior que zero, subtrai um 1 e faz um novo request com a
    versão HTTP atualizada (se o último dígito for zero, sai da função)
    val lesser = if (http.last().code > '0'.code) http.last().code - '0'.code - 1 else return
    makeRequest(client, url, "HTTP/1.$lesser")
  }
}
}
```

Parte da criação do código foi proporcionada ao desenvolvimento de outras mensagens de estado da resposta, entre os quais:

Código do estado HTTP	Funcionalidades	Representação no nosso código
1xx	Informação de transferência de protocolos	1
2xx	Pedido do cliente com sucesso	2
3xx	Necessário ação extra da parte do cliente	3
4xx	Não acessível	4
5xx	Erro do servidor	5

- Mensagem no Terminal:

```
HTTP/1.1 301 Moved Permanently
Server: nginx
Date: Thu, 13 Apr 2023 15:40:03 GMT
Content-Type: text/html
Content-Length: 162
Connection: close
Location: https://www.isel.pt/
```

Figura 3

3. Wireshark

Abrimos o programa Wireshark e filtramos a pesquisa por “TCP”

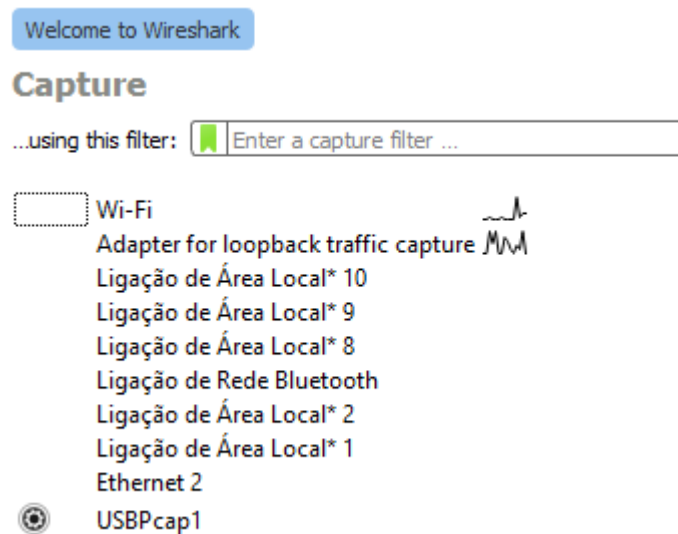
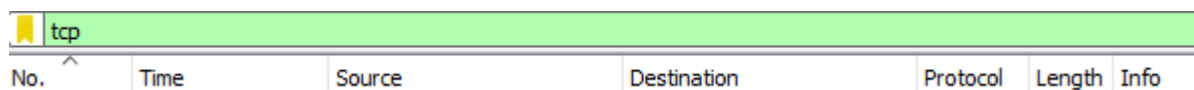


Figura 4

The image shows the top part of the Wireshark packet list window. A green filter bar at the top contains the text 'tcp'. Below the filter bar is a table with the following columns: 'No.', 'Time', 'Source', 'Destination', 'Protocol', 'Length', and 'Info'. The 'No.' column has a small upward arrow icon next to it.

No.	Time	Source	Destination	Protocol	Length	Info
-----	------	--------	-------------	----------	--------	------

Figura 5

Efetuamos a ligação entre o Web Client e o Web Server e este foi o resultado obtido:

No.	Time	Source	Destination	Protocol	Length	Info
61	7.423436	194.210.199.5	193.137.128.195	TCP	66	52405 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1363 WS=256 SACK_PERM
62	7.423596	193.137.128.195	194.210.199.5	TCP	66	80 → 52405 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
63	7.431264	194.210.199.5	193.137.128.195	TCP	54	52405 → 80 [ACK] Seq=1 Ack=1 Win=132096 Len=0
64	7.469367	194.210.199.5	193.137.128.195	HTTP	96	GET / HTTP/1.1
65	7.514478	193.137.128.195	194.210.199.5	TCP	54	80 → 52405 [ACK] Seq=1 Ack=43 Win=524544 Len=0
158	12.483031	193.137.128.195	194.210.199.5	HTTP	298	HTTP/1.1 302 Found
159	12.483443	193.137.128.195	194.210.199.5	TCP	54	80 → 52405 [FIN, ACK] Seq=245 Ack=43 Win=524544 Len=0
160	12.524133	194.210.199.5	193.137.128.195	TCP	54	52405 → 80 [ACK] Seq=43 Ack=246 Win=131840 Len=0
161	12.572638	194.210.199.5	193.137.128.195	HTTP	114	Continuation
162	12.573760	194.210.199.5	193.137.128.195	TCP	54	52405 → 80 [FIN, ACK] Seq=103 Ack=246 Win=131840 Len=0
163	12.573821	193.137.128.195	194.210.199.5	TCP	54	80 → 52405 [ACK] Seq=246 Ack=104 Win=524544 Len=0

Figura 6

Quando é realizado o GET/ HTTP/1.1 a mensagem do código do estado HTTP foi o 302 Found, de seguida foi realizado automaticamente pelo programa um novo request com a localização atual do Web Server.

Em relação ao “Continuation”, esta mensagem significa que o segundo pedido efetuado pelo Web Client foi realizado pelo mesmo utilizador.

No.	Time	Source	Destination	Protocol	Length	Info
35	6.396492	194.210.199.5	192.68.221.35	TCP	66	52561 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
36	6.397814	192.68.221.35	194.210.199.5	TCP	66	80 → 52561 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1363 SACK_PERM WS=128
37	6.397910	194.210.199.5	192.68.221.35	TCP	54	52561 → 80 [ACK] Seq=1 Ack=1 Win=132096 Len=0
38	6.424838	194.210.199.5	192.68.221.35	HTTP	92	GET / HTTP/1.1
39	6.426177	192.68.221.35	194.210.199.5	TCP	54	80 → 52561 [ACK] Seq=1 Ack=39 Win=64256 Len=0
40	6.426603	192.68.221.35	194.210.199.5	HTTP	404	HTTP/1.1 301 Moved Permanently (text/html)
41	6.426603	192.68.221.35	194.210.199.5	HTTP	349	HTTP/1.1 400 Bad Request (text/html)
42	6.426751	194.210.199.5	192.68.221.35	TCP	54	52561 → 80 [ACK] Seq=39 Ack=647 Win=131328 Len=0
43	6.519819	194.210.199.5	192.68.221.35	HTTP	101	Continuation
44	6.521037	194.210.199.5	192.68.221.35	TCP	54	52561 → 80 [FIN, ACK] Seq=86 Ack=647 Win=131328 Len=0
45	6.521125	192.68.221.35	194.210.199.5	TCP	54	80 → 52561 [ACK] Seq=647 Ack=86 Win=64256 Len=0
46	6.522072	192.68.221.35	194.210.199.5	TCP	54	80 → 52561 [ACK] Seq=647 Ack=87 Win=64256 Len=0

Figura 7

No.	Time	Source	Destination	Protocol	Length	Info
174	50.212499	194.210.199.5	192.68.221.35	TCP	66	52799 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
175	50.305014	192.68.221.35	194.210.199.5	TCP	66	80 → 52799 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1363 SACK_PERM WS=128
176	50.305098	194.210.199.5	192.68.221.35	TCP	54	52799 → 80 [ACK] Seq=1 Ack=1 Win=66560 Len=0
177	50.328635	194.210.199.5	192.68.221.35	HTTP	92	GET / HTTP/1.0
178	50.354873	192.68.221.35	194.210.199.5	TCP	54	80 → 52799 [ACK] Seq=1 Ack=39 Win=64256 Len=0
179	50.362353	192.68.221.35	194.210.199.5	HTTP	399	HTTP/1.1 301 Moved Permanently (text/html)
180	50.362353	192.68.221.35	194.210.199.5	TCP	54	80 → 52799 [FIN, ACK] Seq=346 Ack=39 Win=64256 Len=0
181	50.362455	194.210.199.5	192.68.221.35	TCP	54	52799 → 80 [ACK] Seq=39 Ack=347 Win=66304 Len=0
182	50.418915	194.210.199.5	192.68.221.35	HTTP	101	Continuation
183	50.420315	194.210.199.5	192.68.221.35	TCP	54	52799 → 80 [FIN, ACK] Seq=86 Ack=347 Win=66304 Len=0

Figura 8

Fizemos uma segunda conexão do nosso Web Client mas desta vez com o endereço IP do ISEL. A diferença entre ambas figuras 7 e 8 foi o HTTP/1.1 e HTTP/1.0.

Na figura X, as respostas da ligação foram “301 Moved Permanently” e “400 Bad Request”. Na figura Y as respostas foram “301 Moved Permanently” e “Continuation”.

Conclusão

Como demonstrado, através do conteúdo lecionado na cadeira de Redes de Computadores, fomos capazes de estabelecer uma conexão com sucesso com um Web Server utilizando um Web Client.

A realização deste trabalho não só aprimorou o nosso conhecimento da cadeira como também as nossas habilidades de programação e pensamento crítico.

Recursos utilizados:

- Powerpoints disponibilizados pela cadeira Redes de Computadores (Verão) - 2223 no Moodle do ISEL
- IntelliJ, para a realização do código em Kotlin
- XAMPP, para a criação do Web Server
- Wireshark, para observar as conexões entre Web Client e Web Server