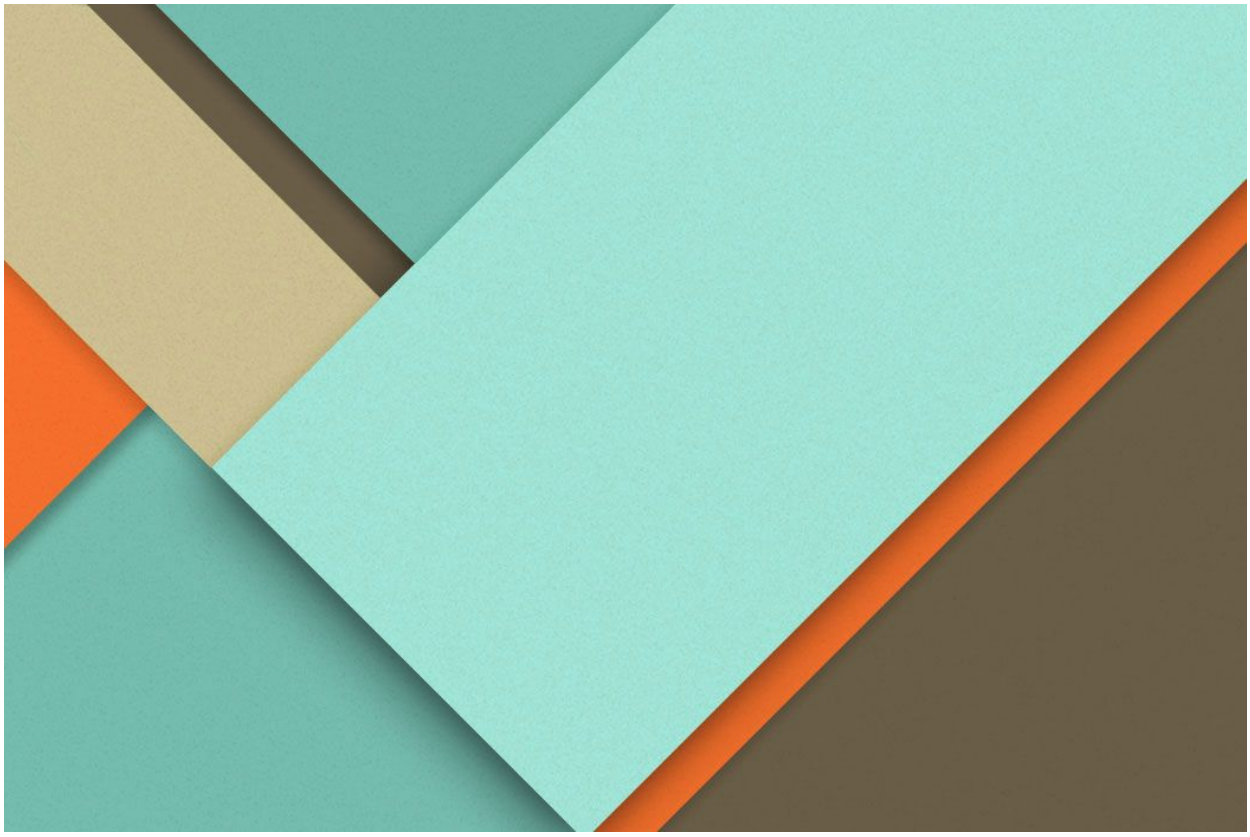
Universidad de Valparaíso
CHILE



Tarea 2

Nombres:

Ariel Salgado

Héctor Delgado

Benjamín San Martín

Profesor:

Gabriel Astudillo

Ayudante:

Pablo Olivares

Introducción

A través de el archivo sockets que el Profesor facilitó en el aula virtual, se buscará modificar el archivo "TCPEchoServer" para crear un servidor iterativo sencillo, que sea capaz de servir páginas web con solo texto. Además se tendrá parametrizado en un archivo de configuración JSON, la dirección IP , puerto TCP, directorio raíz donde se ubican las páginas web, etc.

Más que un trabajo, se podrá experimentar paso a paso como modificar un servidor básico ya hecho, además se podrá entender mejor la funciones creadas y las que la están complementando de una forma escrita como también de una manera Gráfica con el modelo 4+1.

Explicación y documentación del código

1. Cómo uno de los primeros pasos, se ocupó la carpeta de "Socket/echo_example/echoServerSrc" ,y se editó el archivo TCPEchoServer.cc
    a. El primer objetivo fue modificar el código para que muestre solo 1 pagina, así se tendría una noción de cómo funcionaria el servidor eco.

```
28    std::cout << "Handling client ";
29    std::ifstream archivo("holamundo.html");
30    std::string lectura = "";
31    std::string a;
32
33    try {
34      std::cout << sock->getForeignAddress() << ":";
35    } catch (SocketException e) {
36      std::cerr << "Unable to get foreign address" << std::endl;
37    }
38    try {
39      std::cout << sock->getForeignPort();
40    } catch (SocketException e) {
41      std::cerr << "Unable to get foreign port" << std::endl;
42    }
43    std::cout << std::endl;
44    try{
45      sock->send("HTTP/1.1 200 ok\r\nContent-Type: text/html\r\n\r\n",44);
46      if(archivo.is_open()){
47          while(getline(archivo,a)){
48              lectura = lectura + a + "\n";
49          }
50          archivo.close();
51      }
52      else{
53          std::cerr << "Error en lectura." << std::endl;
54          exit(EXIT_FAILURE);
55      }
56    }
57    catch (std::ifstream::failure e) {
58      std::cerr << "Error al abrir el archivo." << std::endl;
59      exit(EXIT_FAILURE);
60    }
61    sock->send(lectura.c_str(),lectura.length());
```

b. Al ya tener claro cómo funciona el código, el resto se centró en buscar la manera de obtener las peticiones que pedía el usuario

```
GET /h1.html HTTP/1.1
```

Esta estaba guardada en la variable “echoBuffer”, de tipo char[], pero para obtener la parte específica que se necesita para diferenciar los “get's”, se tenía que pasar a String.

c. Para pasar a String “echoBuffer” se implementó un algoritmo que permite guardar la petición del navegador Request en la variable “pagina”.

```
53  std::string pagina;
54  char *token = strtok(echoBuffer," ");
55  for(int i = 0; i < 2; i++){
56      if(i == 0){
57          token = strtok(NULL," ");
58      }
59      else{
60          pagina = token;
61      }
62  }
63  //AQUI ESTA LA PAGINA
64  pagina = pagina.substr(1, pagina.length());
```

Específicamente lo que hace es segmentar “echoBuffer” a través de un delimitador “ “(espacio) y lo guarda dentro de un arreglo con puntero. Luego, ya que el Request se encuentra entre los 2 primeros espacios, se considera la segunda ubicación dentro del puntero y se rescata este valor. Finalmente se le quita el “/” a página, así solo quedamos con la petición del navegador.

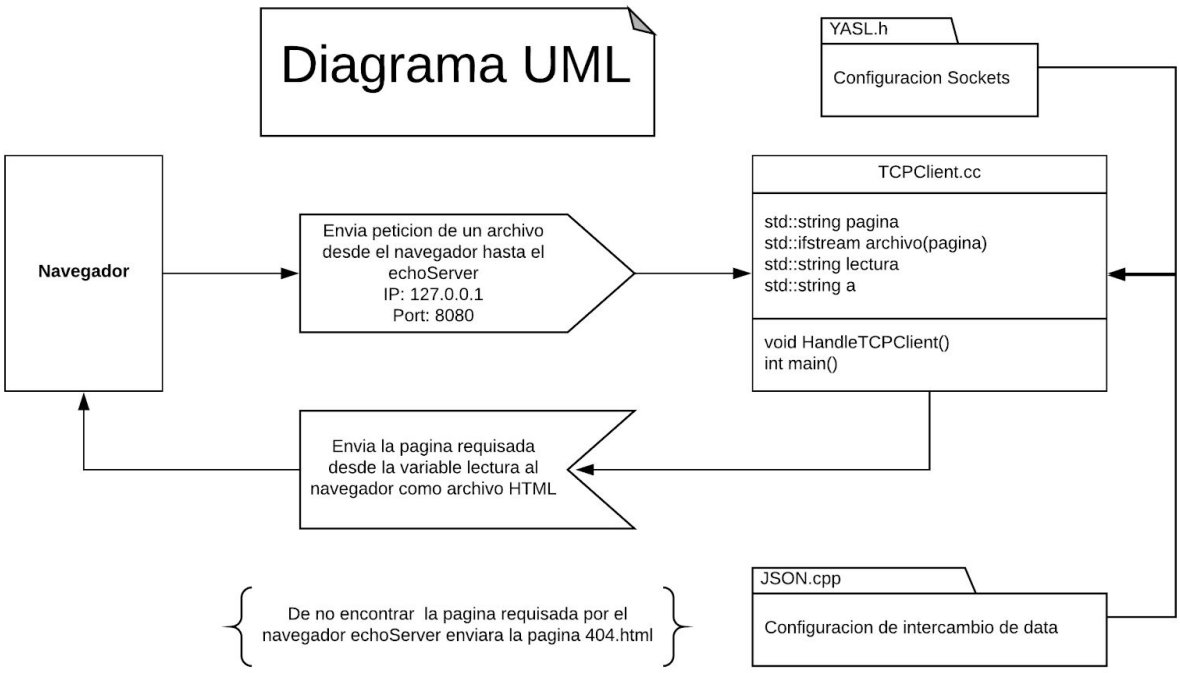
d. Ya que se tiene la petición del Navegador, y solo se enviarán páginas dentro de la carpeta, se crearon 2 páginas más y se hizo una comparación simple con la petición que está dentro de la variable “pagina”, en caso de que la petición no fuera la correcta, se enviará un error 404 indicando al supuesto usuario que la página no existe.

```
66
67  if(pagina == "holamundo.html"){
68      //sock->send(lectura.c_str(),lectura.length());
69  }
70  else if(pagina == "chaomundo.html"){
71      //sock->send(lectura.c_str(),lectura.length());
72  }
73  else if(pagina == "h1.html"){
74      //sock->send(lectura.c_str(),lectura.length());
75  }
76  else{
77      pagina = "404.html";
78      //sock->send(lectura.c_str(),lectura.lenght());
79  }
```

2.- Al ya poder mostrar más de una página, se implementó el archivo json, donde se busca crear una configuración externa del código

```
{
    "ip":"127.0.0.1",
    "puerto":8080,
    "root_dir":"www-data/",
    "notFoundFile":"www-error/404.html"
}
```

Vista lógica



Vista de Procesos