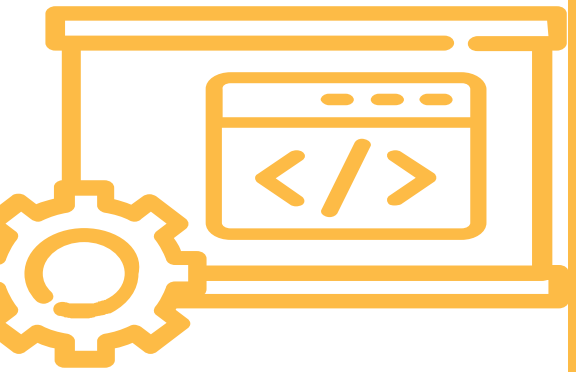


Proyecto Integrador
Etapa II

Juan Aguila Torres
Sebastián Rodríguez Tencio
Gabriel Gonzáles Flores

Temas Abarcados

1. Modificaciones al Cliente
2. Servidor de Piezas
3. HTML

01

Modificaciones al Cliente

HttpClient

```
#ifndef HttpClient_h
#define HttpClient_h

class HttpClient {

    public:
    /**
     * @brief Muestra cuales figuras se encuentran disponibles en el servidor de piezas.
     * @return std::string figures_string con las figuras disponibles.
     */
    void showFigures();

    /**
     * @brief Pide al cliente que digite la figura que desea
     * @return std::string input_string con la figura solicitada por el cliente
     */
    std::string requestFigure();

    /**
     * @brief Extrae de la respuesta del servidor web la lista de piezas de Lego y la muestra en pantal
     * @param html std::string con la respuesta del servidor web.
     */
    void displayLegos(std::string html);

    private:

};

#endif
```

SSLClient

```
#define PORT 5678
#define BUFSIZE 512

int main( int argc, char ** argv ) {
   Socket s('s');     // Crea un socket de IPv4, tipo "stream"
   char buffer[ BUFSIZE + 1];

   HttpClient client;
   // Muestra las figuras que se pueden escoger
   client.showFigures();

   // Solicitar el nombre de la figura al usuario y lo almacena en input_str
   std::string input_string = client.requestFigure();
   char* input = new char[input_string.length() + 1];
   std::strcpy(input, input_string.c_str());
```

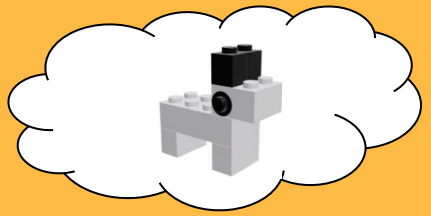
Se despliegan las opciones

Se inicializa SSL para el socket y se realizan distintos métodos SSL

```
s.InitSSL();
char* host =  (char *) "127.0.0.1";
s.SSLConnect( host, PORT ); // Same port as server

if ( argc > 1 ) {
   s.SSLWrite( argv[1], sizeof(argv[1]) );      // Send first program argument to server
} else {
   s.SSLWrite( input, sizeof(input) );
}

int count, itr_count = 0;  // Contador de iteraciones.

// String para almacenar la respuesta del servidor web.
std::string html;

while ((count = s.SSLRead(buffer, BUFSIZE)) > 0) {
   buffer[count] = '\0';
   html += buffer;
   itr_count++;
}
client.displayLegos(html);

delete[] input;
```

02

Servidor de Piezas



Análisis del Servidor de Piezas

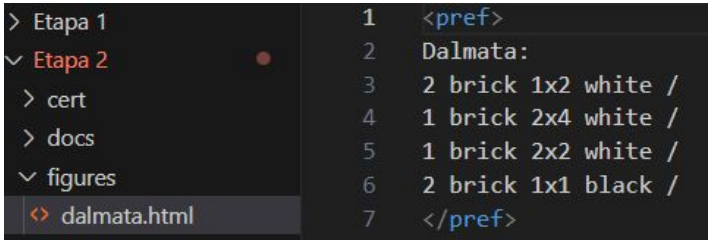
```
void task( Socket * client ) {
   char a[ BUFSIZE ];

   client->SSLAccept();
   client->SSLRead( a, BUFSIZE );   // Read a string from client, data will be limited by BUFSIZE bytes
   std::cout << "Server received: " << a << std::endl;
   std::string htmlName(a);
   htmlName = "figures/" + htmlName + ".html";

   std::ifstream file(htmlName, std::ios::binary);
   if (!file.is_open()) {
      std::cerr << "Error al abrir el archivo" << std::endl;
   }

   std::string html_string;
   std::string line;
   while (std::getline(file, line)) {
      html_string += line;
   }

   file.close();

   char* sendBack = new char[html_string.length() + 1];
   std::strcpy(sendBack, html_string.c_str());

   client->SSLWrite( sendBack, strlen(sendBack) );    // Write it back to client, this is the mirror fu
   client->Close();     // Close socket in parent

}
```

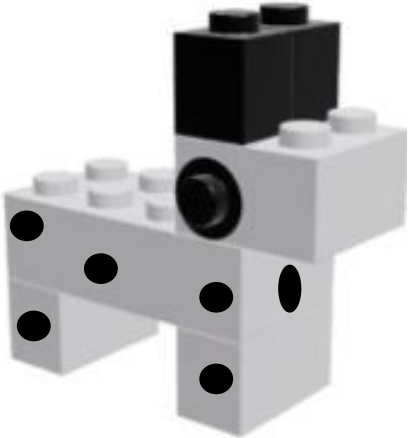
Task

Main

```
int main( int argc, char ** argv ) {
   std::thread * worker;
   Socket * s1, * client;

   s1 = new Socket( 's' );

   s1->Bind( PORT );     // Port to access this mirror server
   s1->Listen( 5 );      // Set backlog queue to 5 conections
   s1->SSLInitServer( "cert/ci0123.pem", "cert/ci0123.pem" );

   for( ; ; ) {
      client = s1->Accept();      // Wait for a client conection
      client->SSLCreate( s1 );
      worker = new std::thread( task, client );
   }

}
```

Figuras - HTML





```cpp
void HttpClient::displayLegos(std::string html) {

    // Inicializar la cantidad total de piezas de Lego en 0.
    int total_quantity = 0;

    // La expresión regular para coincidir con la lista de piezas de Lego.
    std::regex regex(R"((\d+)\s*(brick.*?)\s*\/)");
    // Iterar sobre coincidencias
     std::sregex_iterator it(html.begin(), html.end(), regex);
     std::sregex_iterator end;
     //std::cout << "Hola displayLegos" << html << std::endl;
    while (it != end) {
        // Obtener la coincidencia.
        std::smatch match = *it;
        // Extraiga la cantidad y descripción de la pieza de Lego.
        int quantity = std::stoi(match[1].str());
        total_quantity += quantity;
        std::string description = match[2].str();
        // Generar el resultado
        std::cout << quantity << " " << description << std::endl;
        // Aumentar el iterador.
        ++it;
    }
    // Verificar si se encontraron piezas de Lego.
    if (total_quantity == 0) {
        std::cout << "La figura no existe o no se encontraron piezas de lego para esta figura." << std::e
    } else {
        // Imprimir el total de piezas de Lego.
        std::cout << "Total de piezas para armar esta figura: " << total_quantity << std::endl;
    }
}
```

Gracias!