Universidad de Costa Rica
Facultad de Ingeniería

Escuela de Ciencias de la Computación e informática

Curso CI-0123 PIRO

Proyecto Integrador

Entregable #1

Profesores:
Francisco Arroyo Mora
José Antonio Brenes Carranza

Asistente:
Lester Cordero Murillo

Estudiantes:
(B70257) Ricardo Alfaro Víquez
(B86875) Luis Eduardo Rojas Carrillo
(B80986) Allan Barrantes Chaves
(B67454) Mario Vargas Campos

II Semestre, 2020

1. Introducción

En este primer entregable se presenta una versión simple del programa, en el cual el usuario realizará una consulta sobre la información del COVID en Costa Rica al día actual de la consulta, ya sea filtrando por provincias, o de manera general a nivel país.

El propósito de realizar esta entrega es lograr hacer una conexión correcta mediante un socket y hacer una petición a una página web, además de almacenar de manera correcta los datos recibidos, para posteriormente mostrar una respuesta resumida al usuario, según la consulta realizada.

Para esta entrega fue de gran utilidad el protocolo HTTP, en el cual se exponen generalidades de las consultas http, como el formato de las mismas y los diversos datos de retorno, así como sus códigos de error con su respectivo significado. Fue necesario además comprender correctamente el funcionamiento de las clases Socket, cliente y servidor que se habían entregado en las tareas cortas del curso, pues fueron una base clave para la correcta implementación de las solicitudes mediante socket, y el manejo de las respuestas.

En cuanto a la parte del parseo de los documentos recuperados, se investigó acerca del manejo de los archivos tipo csv y las expresiones regulares, así como la correcta impresión en pantalla.

2. Objetivo General

A continuación se establece el objetivo general del proyecto:

Implementar un programa que permita conectarse a un servidor con el fin de obtener datos actualizados sobre el COVID-19 y mostrarselos al usuario.

3. Objetivos Específicos

A continuación se establecen los objetivos específicos para esta etapa del proyecto:

- Construir un socket funcional que permita hacer conexiones exitosas a un servidor
- Realizar una consulta a un servidor mediante un llamado de socket utilizando el protocolo HTTP.
- Recibir la respuesta del servidor y parsear de manera correcta los datos necesarios para mostrarlos al usuario.

4. Descripción de la Solución

Primeramente se implementó la clase socket, la cual permite establecer una conexión a un servidor mediante una dirección IP y un puerto específico. Para este caso se utilizó el puerto 80, ya que es el puerto que se utiliza comúnmente para este tipo de consultas. Mediante el socket, se realizó una conexión a http://geovision.uned.ac.cr/oges/ para consultar los datos actualizados sobre el covid en Costa Rica, utilizando el protocolo HTTP como formato para hacer la solicitud. Dado que los datos se encontraban en diferentes archivos, fue necesario realizar 5 consultas de archivos diferentes, para obtener la información completa. (Fallecidos, Recuperados, Activos, Positivos y un informe General).

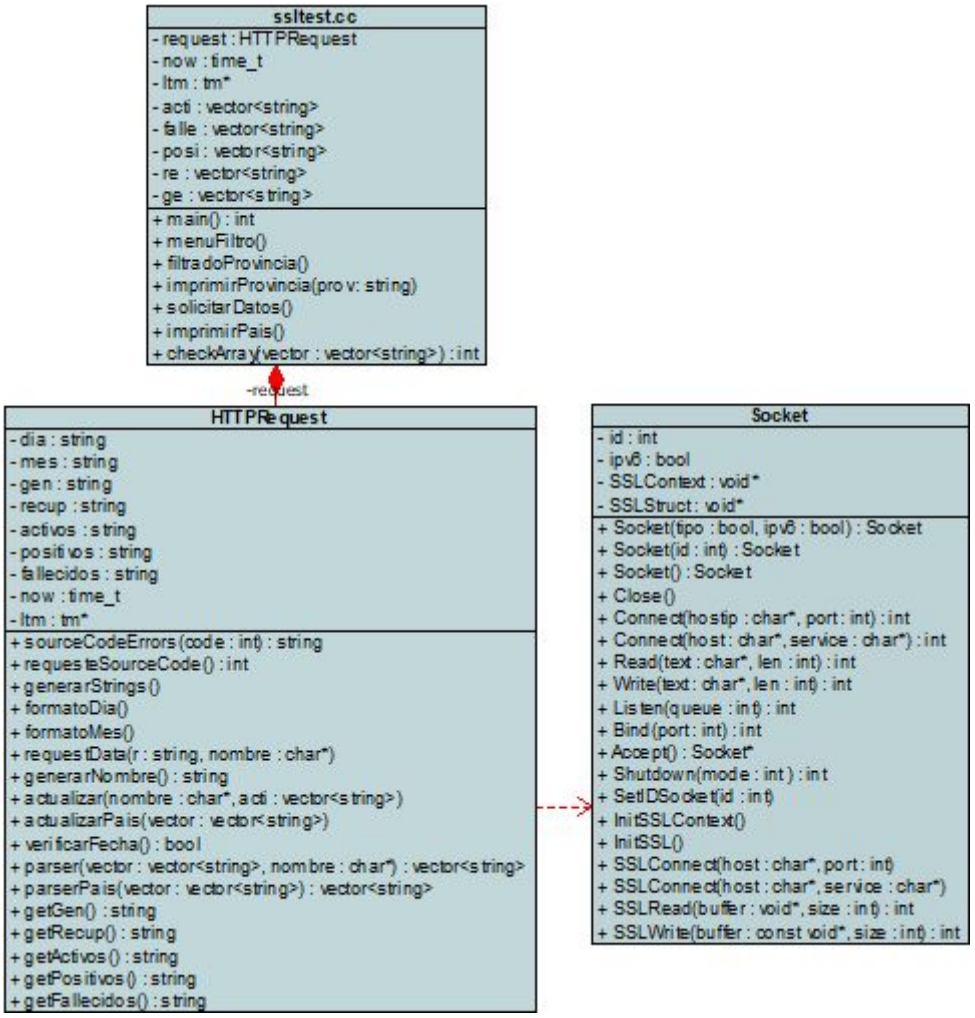
Una vez obtenida la respuesta del servidor, se procedió a parsear los datos y almacenarlos en un archivo .csv (el archivo general se guardó en un documento llamado Pais.csv, mientras que los demás datos fueron guardados todos juntos en un solo documento llamado Actualizado.csv).

Finalmente se abrieron una vez más los archivos, para leerlos y mostrarlos de manera tabulada al usuario, según la consulta que este hubiese realizado.

Manejo de Errores: Con respecto al manejo de errores se implementó una funcionalidad, en la que el programa al solicitar datos al servidor, mediante el método *HEAD*, recibe el source code asociado a la página web. En caso de obtener un *200 OK*, el programa obtiene los datos de manera correcta. Sin embargo, en caso de obtener un *400, 404, 501* ó *505*, devuelve el mensaje de error asociado al usuario, y por lo tanto se dirige a la base de datos a leer un archivo del día anterior. Estos 4 código de error serán los que se manejan durante esta entrega, posteriormente se añadirán otros.

5. Esquema de la Solución

En el siguiente UML, se muestra la conformación de las clases del proyecto y cómo interactúan entre ellas. Además, se especifican las variables o los métodos propios de cada clase.



6. Arquitectura de la Solución

La siguiente tabla, muestra la arquitectura del proyecto. Se define la conformación de las clases, y la funcionalidad principal para cada una de ellas. Además, de librerías o frameworks que destacan en la implementación del código.

Concepto	Descripción
Socket.h	Esta clase define los métodos y los atributos correspondientes a la clase Socket.cc. Librerías: -OpenSSL: Esta librería permite el uso de herramientas para el manejo de protocolos SSL/TLS.
Socket.cc	Esta clase efectúa la comunicación con el servidor. Mediante una solicitud, permite leer y escribir, esto con el fin de obtener datos provenientes de un servidor a partir de dicha solicitud.
HTTPRequest.h	Esta clase define los métodos y los atributos correspondientes a la clase HTTPRequest.cc. Librerías: -Fstream, iostream: Son librerías que permiten herramientas para el manejo de I/O dentro de cualquier programa.
HTTPRequest.cc	Esta clase realiza un manejo de los request que se realizarán al servidor para solicitar los datos. Realiza un manejo de los archivos .csv, en los cuales se parsean los datos solicitados. Realiza un manejo de errores de los source code siempre que se soliciten los datos mediante un HTTP Request.
ssltest.cc	Esta es la clase main del programa. Maneja los menús que se le desplegarán al usuario para la interacción con el mismo. Despliega los resultados tabulares que el usuario solicite. Librerías: -Fstream, iostream: Son librerías que permiten herramientas para el manejo de I/O dentro de cualquier programa. -Socket.h, HTTPRequest.h: Incluye ambas clases para el manejo del programa.

7. Limitaciones de su Solución

A pesar de los intentos, no se logró generar un Makefile unificado con los archivos en diferentes carpetas, pues por las características de los Makefile, no arrastran automáticamente los directorios de las librerías, por lo cual se presentaba un error al intentar generar el archivo Socket.o, dado que las librerías no se encontraban.

8. Pruebas de funcionalidad

Describa los casos de prueba para esta etapa del proyecto y sus resultados esperados. Utilice tablas cuando requiera indicar que se necesita ejecutar una instrucción en terminal.

Caso#1: Muestra la provincia de Alajuela.

```
Elija alguna de las opciones del menu
1. Ver opciones de Provincias
 2. Ver casos generales del pais
 3. Salir del programa
1
Elija una de las provincias
 1. San José
 2. Alajuela
 3. Limón
 4. Puntarenas
 5. Guanacaste
 6. Cartago
 7. Heredia
 8. Volver al menú principal
 9. Salir del programa
```

Provincia	Cantón	Positivos	Activos	Recuperados	Fallecidos
Alajuela	Alajuela	3778	3117	624	37
Alajuela	Atenas	178	93	82	3
Alajuela	Grecia	438	289	141	8
Alajuela	Guatuso	160	37	121	2
Alajuela	Los Chiles	284	60	222	2
Alajuela	Naranjo	469	292	175	2
Alajuela	Orotina	90	39	51	0
Alajuela	Palmares	274	136	137	1
Alajuela	Po◆s	265	87	176	2
Alajuela	R◆o Cuarto	185	156	29	0
Alajuela	San Carlos	1555	944	604	7
Alajuela	San Mateo	12	5	7	0
Alajuela	San Ram◆n	568	217	346	5
Alajuela	Sarch◆	80	35	45	0
Alajuela	Upala	188	42	143	3
Alajuela	Zarcero	109	81	28	0

Caso#2: Muestra los datos generales de Costa Rica.

```
2
País      Posi      +Posi      Falle      +Falle      UCI      +UCI      Recup      +Recup      Muestras      Activos

CR        53969      1420       583        16         221       19       20710      388         178446        32676

Elija alguna de las opciones del menu
1. Ver opciones de Provincias
 2. Ver casos generales del pais
 3. Salir del programa
```

Caso#3: Muestra manejo de error HTTP 404, archivo no encontrado.

```
2
El archivo correspondiente al dia de hoy no se encuentra disponible
Se procedera a mostrar un archivo con la fecha anterior mas reciente

País      Posi      +Posi      Falle      +Falle      UCI      +UCI      Recup      +Recup      Muestras      Activos

CR        53969      1420       583        16         221       19       20710      388         178446        32676
```

Caso#4: Muestra manejo de menú en caso de ingreso de información inválida.

9. Protocolo HTTP

Primeramente especificar un formato de solicitud de recursos en el servidor que va a existir, la idea es asimilar el comando GET de HTTP, en el cual en nuestra opinión debería solicitarse de la siguiente manera: GET recurso/HTTP/1.1/host: servidor.piro. Seguido de este request se recibirá un código de respuesta para saber si el request de información es válido o si la información está disponible. Si el request es válido entonces extraerá la información del servidor y estará a la disposición del cliente.

Volviendo al inicio con lo del request, para cuando se escribe el recurso se deben analizar casos especiales como nombres con tildes o espacios. Para casos con tildes, proponemos que se escriba el nombre sin ninguna tilde y en los casos que hayan espacios que se representen los espacios con un guión bajo (_).

Los archivos que se recibirán si el request es válido y que estarán almacenados en los servidores serán de tipo csv para facilitar el acceso a los datos.

10. Valoración

Valoración Proyecto	Primera Etapa	Nota Proyecto
Miembro del Equipo	**Nota Individual**	100
Ricardo Alfaro Víquez	100	
Allan Barrantes Chaves	100	
Luis Rojas Carrillo	100	
Mario Vargas Campos	100	

11. Bitácora

Fecha	Trabajo realizado
8/27/2020	Preparamos diagramas en lucidchart y analizamos cómo resolver el problema
8/28/2020	Clase de PIRO comenzamos a realizar las conexiones del socket con el servidor de datos de la UNED
9/7/2020	Se optimizó la conexión del socket con servidor de datos de la UNED para obtener archivos csv , se creó el git y archivos como gitignore, también los branches para entrega y los de cada developer
9/8/2020	Se creó parser para obtener los datos de los archivos csv del sitio de la UNED
9/9/2020	Se añadió código para todo lo relacionado a protocolo HTTP principalmente códigos de error para saber cuando ir al servidor a solicitar los datos y cuando no. Se comenzaron a realizar pruebas para el filtrado de los datos solicitados por un cliente
9/10/2020	Se realizó depuración del código para que el filtrado de los datos funcione bien, poder obtener los datos de la fecha actual y en caso de que aún no existan los archivos aún, entonces proceder a obtener un archivo del día anterior. Comenzamos documentación
9/11/2020	Finalizamos documentación y depuración completa del código.

12. Referencias Bibliográficas

[1] Contributors, MDN. (2020). HTTP response status codes. Septiembre 8, 2020, de MDN Web Docs. Sitio web: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status.

[2] Cplusplus.com. (2020). General C++ Programming. Septiembre 8, 2020, de cplusplus.com. Sitio web: http://www.cplusplus.com/forum/general/.

[3] OpenSSL Software Foundation. (1997-2018). Open SSL Cryptography and SSL/TLS Toolkit. Septiembre 7, 2020, de OpenSSL Software Foundation. Sitio web: https://www.openssl.org.