Universidad de Costa Rica
Facultad de Ingeniería

Escuela de Ciencias de la Computación e Informática

Curso CI-0123 PIRO

Proyecto Integrador

Entregable #4

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 el cuarto entregable, se realiza una integración de las etapas entregadas anteriormente, dejando de lado el sistema operativo NachOS, para retomar nuevamente las particularidades de la comunicación entre sockets, en esta ocasión abordamos la utilización de sockets de tipo datagram para paso de mensajes, así como la implementación correcta del protocolo PIRO para la correcta inicialización de los servidores

A cada grupo de trabajo del proyecto integrador se le asignó una región en particular, la cual contiene un grupo de países repartidos según la geografía de la región asignada, en el caso del grupo 3, corresponde la región de **eastern europe** (europa oriental), que según la información brindada, contiene a los siguientes países: Belarus, Bulgaria, Czechia, Hungary, Poland, Republic of Moldova, Romania, Russian Federation, Slovakia, Ukraine, Albania, Andorra, Bosnia and Herzegovina, Croatia, Gibraltar, Greece, Holy See, Italy, Malta, Montenegro, North Macedonia, Portugal, San Marino, Serbia, Slovenia, Spain. El objetivo de este entregable. es poder hacer consultas sobre los estados de los países de las diferentes regiones, para ello es necesario el manejo de una tabla con los datos de cada servidor, en el cual se almacena la IP en la que se encuentra “escuchando” cada servidor. Esta comunicación inicial se realiza utilizando sockets UDP que envían un broadcast a los servidores disponibles de la red.

Posterior a esto y cuando ya las tablas poseen la información necesaria, se realizan solicitudes por medio de sockets TCP utilizando el protocolo HTTP, momento en el cual cada servidor verifica a cual IP pertenece la región solicitada, para transmitir la solicitud a el servidor ubicado en esa IP, este responde a ese servidor intermedio y finalmente llega la respuesta al cliente.

Esta última entrega recopila todas las habilidades desarrolladas en las entregas anteriores, mostrando la versatilidad de los sockets, de los diversos protocolos para paso de mensajes, y de la importancia de entender su funcionamiento en el desarrollo de aplicaciones con conexiones de red.

2. Objetivo General

A continuación se establece el objetivo general del proyecto:

Implementar un programa que permita mediante un cliente, solicitar datos a un servidor intermedio, el cual se comunica con un servidor de datos mediante el protocolo PIRO para obtener información respecto a un país en una región determinada hasta responder dicha solicitud de nuevo hasta el cliente.

3. Objetivos Específicos

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

- Construir un servidor intermedio que reciba solicitudes de un cliente.
- Realizar una consulta a un servidor de datos mediante un servidor intermedio y viceversa, a través del protocolo PIRO.
- Construir un servidor de datos que provea información respecto a países de una región determinada.

4. Descripción de la Solución

En este último entregable, se implementó lo establecido dentro del protocolo PIRO para enviar y recibir mensajes a través de broadcast para los diferentes países y las regiones a las que estos correspondan. Para la solución utilizamos las tres computadoras de la ECCI asignadas al grupo por los profesores.

Cliente:

Para el cliente no se realizaron muchos cambios, simplemente se realizó la conexión hacia la IP que corresponde a la del servidor intermedio, también se dio formato a las solicitudes para que fuera de acorde al protocolo PIRO, es decir que en este caso desde consola solicitamos al usuario la región y país sobre los cuales desea obtener los datos del covid, el cliente concatena esos dos datos a un string que posee el formato: **region=nombre-region&pais=nombre-pais**, este string se envía al servidor intermedio y el cliente entonces espera por una respuesta.

Servidor intermedio:

Para el servidor intermedio fue necesario manejar varios elementos. Primero se creó una estructura que almacena un string para la región y otro para la IP, esto se hizo para posteriormente crear un array de ese tipo de estructura y así poder simular la tabla en la cual se almacenan los datos (región, IP) de los servidores de datos de los otros grupos cuando estos envían un broadcast hacia nuestro servidor intermedio. Además, fue necesario crear un método para manejar el aspecto de enviar y recibir los broadcast por UDP, este método lo que hace es que si el servidor intermedio se levanta primero que el de datos entonces envía un broadcast a los de datos con su información y así los de datos que estén activos envían una respuesta de vuelta con su IP+región, entonces el servidor intermedio procede a registrar esa información en la tabla en caso de que no se haya registrado aún y así se sabe a cuáles regiones (grupos) se pueden hacer solicitudes. Igualmente, en caso de que un servidor de datos se levante primero y envíe el broadcast, el servidor intermedio lo va a recibir y registrar la información de inmediato y en caso de que se reciba información de otro servidor de otro grupo pero intermedio, simplemente se ignora.

En el servidor intermedio también se hizo código para parsear bien la solicitud, principalmente en el caso de que se haga una solicitud desde un browser. Lo anterior porque si se hace una solicitud desde un browser vendría algo como **GET /?region=region&pais=pais etc…** entonces fue necesario distinguir si era una solicitud de browser para poder pasarle al servidor de datos simplemente el string de la solicitud ya "limpio" (region=region&pais=pais).
Para el caso de los cantones, tomando en cuenta que estas solicitudes son siempre hacia nuestro servidor de datos entonces decidimos utilizar el formato de PIRO pero adaptándolo. Entonces el formato que definimos fue el siguiente: **country=costa-rica&canton=nombre-canton,** para el nombre del cantón lo definimos en minúscula, si hay espacios estos se representan por guión y sin tildes.

Una vez que ya se ha parseado la solicitud del cliente, entonces se define si es de un país y lo que hacemos es sacar el nombre de la región, buscar si está en la tabla y en caso de que si esté, entonces enviamos el request al servidor de datos correspondiente. Si es un cantón solamente enviamos el request a nuestro servidor de datos.

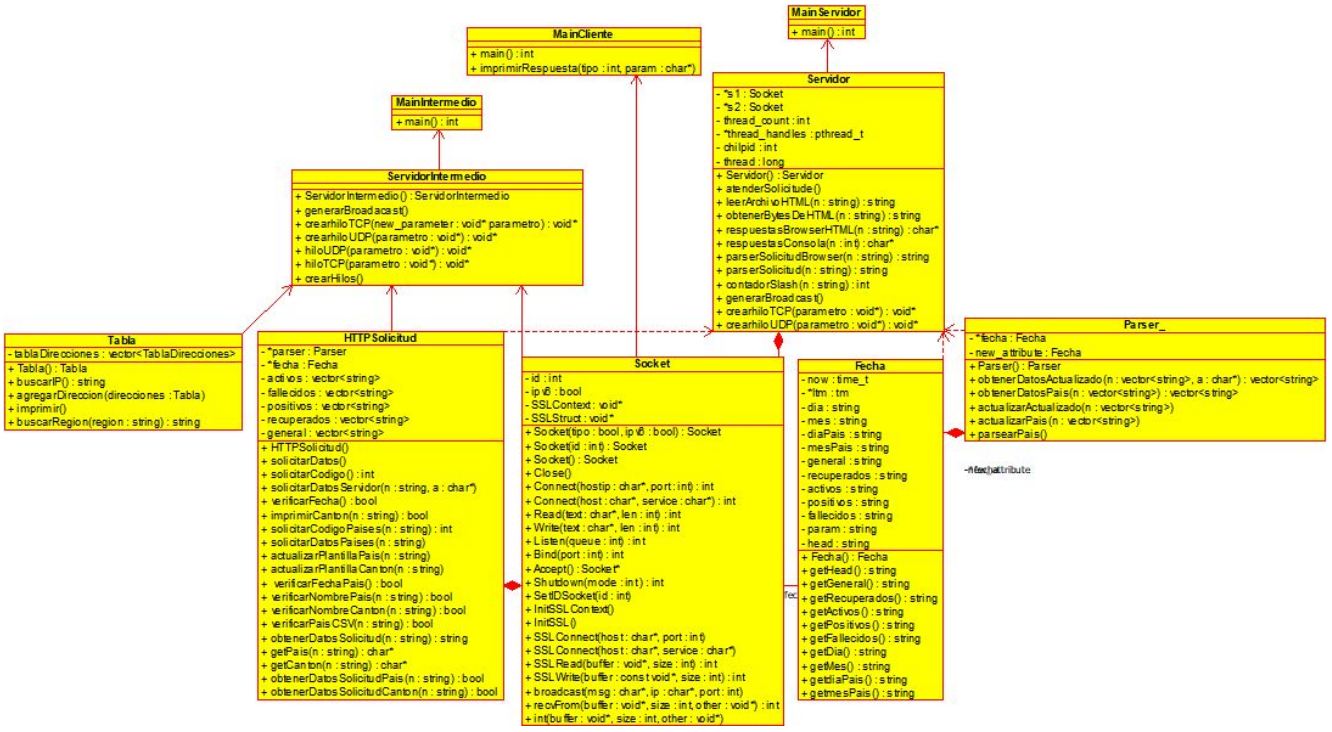
Servidor de datos:

Para el servidor de datos, se creó un método encargado de manejar el broadcast UDP hacia los servidores intermedios con sus datos (1+región+IP) cuando este se levanta o en el caso de que se reciba un broadcast de un servidor intermedio entonces igualmente responder de vuelta con esos mismos datos. Además, fue importante readaptar un poco el código de entregas pasadas encargado de manejar los request para poder parsear con el nuevo tipo de formato que se estableció en el protocolo PIRO para las solicitudes. Lo que hacemos es verificar primero si es un país o un cantón lo que se solicita, si es un país entonces verificamos si corresponde a la lista de países correspondiente a nuestro servidor de datos que es la región de **eastern-europe,** si es así entonces hacemos el request de los datos y los envíamos de vuelta al servidor intermedio que lo solicitó. En el caso de ser un cantón entonces simplemente obtenemos la información y la envíamos de vuelta a nuestro servidor intermedio.

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.

Esquema del Servidor:



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.
HTTPSolicitud.h	Esta clase define los métodos y los atributos correspondientes a la clase HTTPSolicitud.cc. Librerías: -Fstream, iostream: Son librerías que permiten herramientas para el manejo de I/O dentro de cualquier programa.
HTTPSolicitud.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.
MainCliente.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.
Fecha.h	Esta clase define los métodos y los atributos correspondientes a la clase Fecha.cc. Librerías: -Fstream, iostream: Son librerías que permiten herramientas para el manejo de I/O dentro de cualquier programa.

Fecha.cc	Esta clase realiza un manejo de los formatos de fecha para el resto de clases que requieran de su implementación.
Parser.h	Esta clase define los métodos y los atributos correspondientes a la clase Parser.cc. Librerías: -Fstream, vector, ctime, algorithm: Estas librerías permiten el uso de objetos vector, manejo de variables de tiempo y fecha, así como el manejo de archivos.
Parser.cc	-Esta clase realiza un manejo de los datos del proyecto para así realizar los parser necesarios. -Modifica archivos .csv y plantillas HTML necesarias para el almacenamiento y visualización de los datos.
Servidor.h	Define los métodos y los atributos correspondientes de la clase Servidor.cc. Librerías: -Fstream, iostream: Son librerías que permiten herramientas para el manejo de I/O dentro de cualquier programa. -Pthread: Es una librería que permite el manejo de hilos, indispensable para el manejo de sockets.
Servidor.cc	Esta clase realiza un manejo del programa con una función de servidor de datos. Ejecuta solicitudes de un servidor de datos y se los provee al cliente cuando este lo solicite.
MainServidor.cc	Esta clase permite que el servidor se ejecute.
Servidor Intermedio.c	Clase que contiene las clases necesarias para el funcionamiento correcto del servidor intermedio.
Main Intermedio.c	Main que controla el servidor intermedio
Main cliente.c	Clase controladora que maneja un cliente que se conecta al servidor intermedio y servidor de datos.

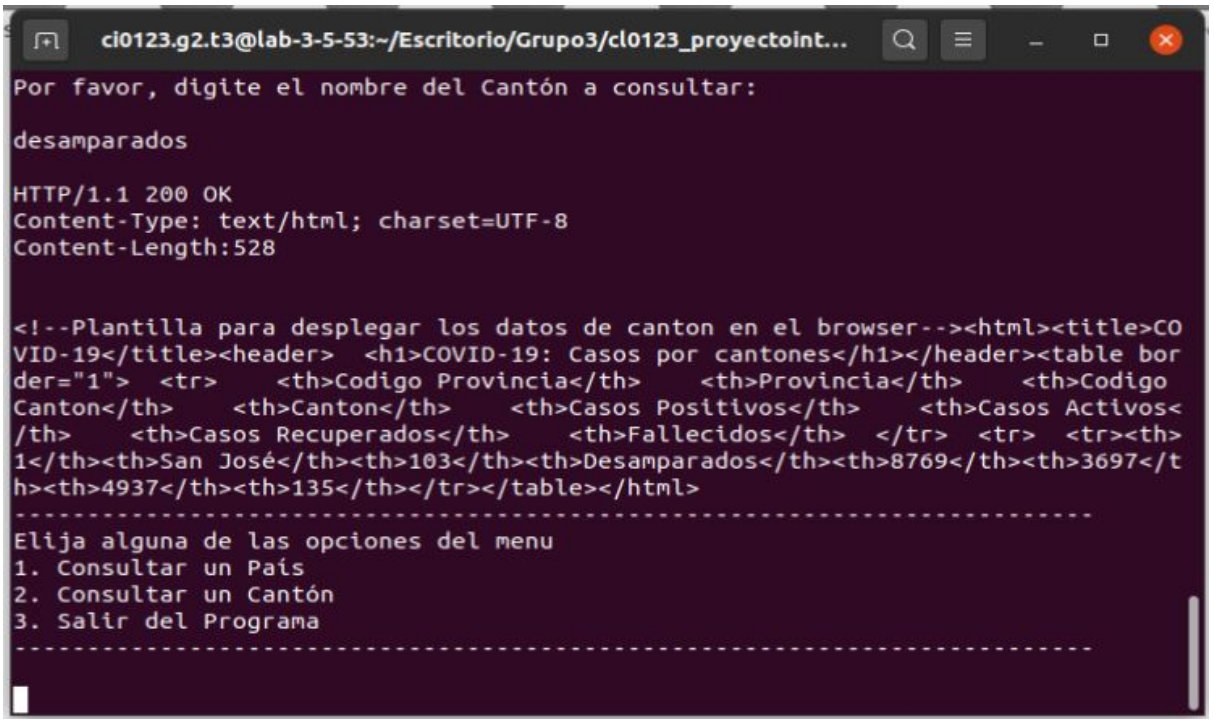
7. Limitaciones de su Solución

A pesar de que se logró la conexión exitosa con el servidor de datos propio utilizando de manera correcta los protocolos tanto PIRO como HTTP, al momento de intentar hacer pruebas con los servidores de otros grupos, logramos conectarnos exitosamente con sus servidores de datos mediante el broadcast y recibir sus respuestas con las respectivas IPs que se almacenaron en la tabla de direcciones, pero no se logró realizar una consulta de país de manera efectiva.

El resultado obtenido fue enviarles la consulta del país de manera correcta al servidor correspondiente, mas no obtener respuesta alguna por parte de ese servidor; siendo una posible razón de este fallo la diferencia de formatos en cuanto a las consultas. Debido a razones de integración de grupos (No todos los grupos tenían sus servidores arriba al mismo tiempo), las pruebas realizadas no fueron suficientes para detectar la causa exacta del error.

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.



```
--------------------------------------------------------------------------
Elija alguna de las opciones del menu
1. Consultar un País
2. Consultar un Cantón
3. Salir del Programa
--------------------------------------------------------------------------

1

Por favor, digite el nombre de la región a consultar:


eastern-europe
Por favor, digite el nombre del país a consultar:

spain

HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Content-Length:816


<!--Plantilla para desplegar los datos de País en el browser--><html><title>COVID-19</title><header>  <h1>COVID-19: Casos por país</h1></head
er><table border="1">  <tr>    <th>Pais</th>    <th>Casos Totales</th>    <th>Nuevos Casos</th>    <th>Fallecimientos Totales</th>    <th>Nue
vos Fallecimientos</th>    <th>Recuperados Totales</th>    <th>Nuevos Recuperados</th>    <th>Casos Activos</th>    <th>Casos Criticos</th>
  <th>Casos Totales/1M</th>    <th>Fallecimientos Totales/1M</th>    <th>Pruebas Totales</th>    <th>Pruebas Totales/1M</th>    <th>Poblacion
 Total</th>  </tr>  <tr><th>Spain</th><th>1,699,145</th><th>+5,554</th><th>46,252</th><th>+214</th><th>N/A</th><th>N/A</th><th>N/A</th><th>2,
371</th><th>36,336</th><th>989</th><th>22,992,742</th><th>491,692</th><th>46,762,516</th></tr></table></html>
--------------------------------------------------------------------------
```

COVID-19: Casos por país

Pais	Casos Totales	Nuevos Casos	Fallecimientos Totales	Nuevos Fallecimientos	Recuperados Totales	Nuevos Recuperados	Casos Activos	Casos Criticos	Casos Totales/1M	Fallecimientos Totales/1M	Pruebas Totales	Pruebas Totales/1M	Poblacion Total
Spain	1,699,145	+5,554	46,252	+214	N/A	N/A	N/A	2,371	36,336	989	22,992,742	491,692	46,762,516

9. Protocolo PIRO

Para la última entrega del proyecto, se definirá un nuevo protocolo para la comunicación entre los servidores intermedios y los servidores de datos. Para la dinámica, se requerirá un uso híbrido entre el protocolo HTTP y el protocolo PIRO. Dentro de las funcionalidades generales que se implementarán, se encuentran las siguientes:

- El sistema se compone de 3 personajes importantes: Los clientes, los servidores intermedios y los servidores de datos.
- Se requiere del uso de 2 protocolos de comunicación, HTTP y PIRO.
- Un cliente puede comunicarse únicamente a un solo servidor intermedio correspondiente.
- Un servidor intermedio puede comunicarse con diversos servidores de datos mediante un broadcast.
- Los servidores de datos responden a los servidores intermedios que les realizaron la solicitud.
- La solicitud entre un servidor intermedio y un servidor de datos se realiza mediante el siguiente formato:

  URL: región=”nombre-región”&pais=”nombre-país”

- El formato para manejar la solicitud requiere el uso únicamente de mayúsculas, los espacios se deben cambiar por guiones intermedios y los nombres de los países deben manejarse en inglés.
- El puerto para comunicarse mediante el protocolo HTTP es el 51000 y el puerto para la comunicación mediante el protocolo PIRO es el 50000.
- Los servidores intermedios se identifican con un Id = 1 y los servidores de datos se identifican con un Id = 0.

A continuación se especifican las funcionalidades de cada personaje dentro del sistema:

<u>Cliente</u>

Un cliente es capaz de solicitarle datos a un servidor intermedio acerca de un país en específico. Esta comunicación se realiza mediante una solicitud del protocolo HTTP y un cliente se puede comunicar únicamente con un único servidor intermedio.

<u>Servidor Intermedio</u>

Un servidor intermedio es el encargado de realizar una funcionalidad tipo proxy entre un cliente y un servidor de datos. Este servidor recibe peticiones por parte de un cliente y a partir de esto, el servidor se comunica con un servidor de datos para obtener dicha información y brindarle al cliente. Este debe recibir la

petición HTTP del cliente y enviarle la misma petición al servidor de datos. Este servidor intermedio almacena una tabla con la información de los servidores de datos. Aca se almacenan las regiones asignadas a cada servidor de datos, por lo que se registran de la siguiente manera:

"región+IP", siguiendo las mismas reglas mencionadas anteriormente.

La comunicación entre un servidor intermedio y un servidor de datos se da de la siguiente manera:

- Se levanta el servidor intermedio.
    - Enviar broadcast a todos los servidores para avisar que estoy activo. Este envío se realiza usando el siguiente formato: string separado con coma que tenga un 1 + [IPv4]. Ej: "1,172.16.4.205".
    - Al enviar el broadcast, pueden darse los siguientes casos:
        - No hay otros servidores de datos activos, por lo que no hay respuestas para procesar y la tabla de servidores de datos queda en su estado actual.
        - Hay otros servidores de datos activos, por lo que se espera una respuesta que contenga un string con el siguiente formato: "Id, IP, region".
        - Hay servidores intermedios activos, se reciben de 1 a N respuestas en el siguiente formato de string: "1,ipv4", pero no se procesa la respuesta.
    - Se deja un oído activo para escuchar los broadcast de otros servidores. Si es un servidor de datos, se agrega a la tabla, pero si es un servidor intermedio, se descarta.

Servidores de Datos

Un servidor de datos es el encargado de recibir la petición HTTP proveniente del servidor intermedio, y a partir de esto, ir a solicitar los datos de la petición y enviarle dichos datos de regreso al servidor intermedio. Estos servidores de datos pueden recibir peticiones de múltiples servidores intermedios. Cada servidor de datos tiene asignado una región de países, los cuales serán solicitados por el cliente.

La comunicación entre un servidor de datos y un servidor intermedio se da de la siguiente manera:

- Se levanta un servidor de datos.
    - Enviar broadcast para avisar al resto de servidores que estoy activo. El formato del envío se da mediante un string de la siguiente manera: "0, [IPv4], región. La región se maneja como un string, en minúscula, y los espacios se manejan con dash.
    - Se deja un oído activo para escuchar los broadcast de otros servidores. Si es un servidor de datos, se descarta, pero si es un servidor intermedio, se espera por la confirmación de la solicitud.

A continuación se presenta un esquema del sistema completo anteriormente mencionado:

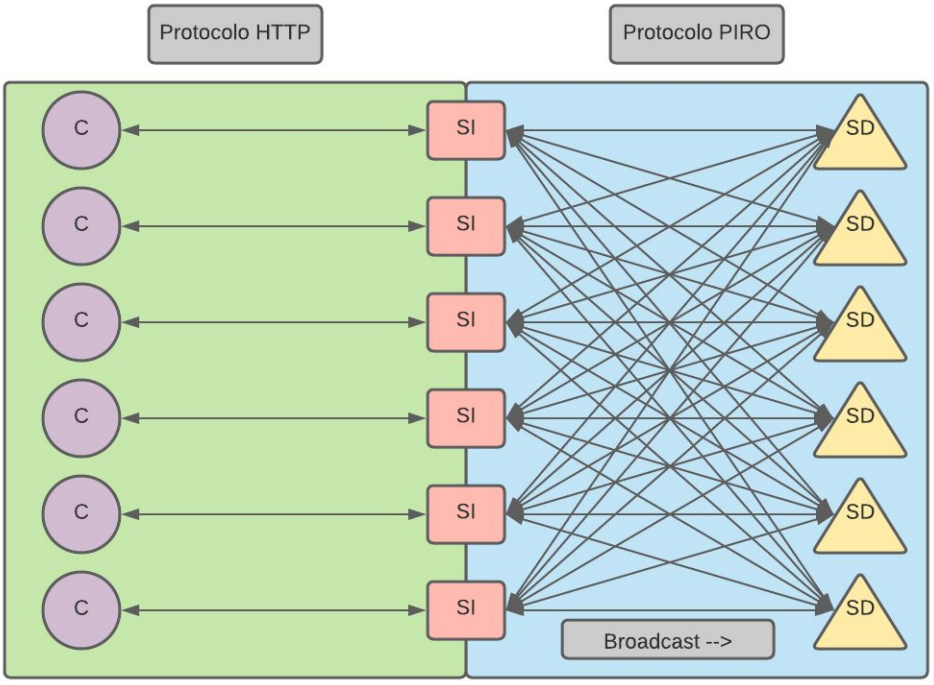


Tabla de Servidores de Datos

Se almacenan las regiones mediante las respuestas de los servidores de datos.

Formato: region + IP

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
11/13/2020	Se empezó a modificar las estructuras del antiguo servidor de datos, para poder utilizar metodos del mismo en la nueva clase del servidor intermedio
11/17/2020	Se creó la clase de servidor intermedio, y se definió el manejo de la estructura tabla
11/20/2020	Se confirmó con los profesores los puertos predeterminados para el servidor de datos e intermedio para el protocolo PIRO
11/25/2020	Se modifico los parametros que pasaba la clase cliente desde consola, para que se adaptara al parseo que se realizaba en el servidor intermedio
11/27/2020	Por cuestiones de implementacion se decició convertir la estructura tabla en un clase, para poder accesar de manera global y utilizarla en los diferentes hilos
12/2/2020	Se corrigieron errores de parseo, así como errores de compilacion
12/3/2020	Se corrigieron problemas que incurrian en el fallo de broadcast desde los distintos servidores
12/4/2020	Se realizaron pruebas entre los servidores intermedios y de datos, de manera local, con los cuales funcionaba el protocolo piro y la solicitud de paises y cantones, tanto desde consola como desde navegador

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.

[4] https://github.com/seleznevae/libfort