

**Universidad de Costa Rica**  
**Facultad de Ingeniería**



**Escuela de Ciencias de la**  
**Computación e informática**

**Curso CI-0123 PIRO**

**Proyecto Integrador**  
Entregable #2

**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 segundo entregable se elevó la complejidad del proyecto al incorporar la capacidad de hacer consultas desde un navegador web, utilizando un formato de consulta específico, además de añadirle la opción de consultar los datos sobre las estadísticas de coronavirus para prácticamente todos los países existentes.

Para ello fue necesario reestructurar casi por completo el programa que se había creado para la entrega anterior, pues nuestro pasado cliente ahora debía ser capaz de fungir como servidor, para atender consultas

Se necesitó también, fortalecer la clase que se encargaba de filtrar los datos correctamente, manejando las diferentes particularidades que involucra la búsqueda por ingreso de texto en vez de la utilización de menú, así como las características de los países y sus nombres, esto modificó también el manejo de errores, pues había que valorar la situación de cuando no se encontraba la consulta pues los datos ingresados del usuario eran inválidos.

También se hizo necesario poder identificar cuándo la consulta venía desde un navegador y cuándo de un cliente en consola, para darle el formato adecuado a la respuesta del servidor y así los clientes pudieran interpretar los datos correctamente.

Al estar usando información de páginas diferentes, con formatos distintos, estos también requerían de un tratamiento particular para su correcto parseo y almacenamiento.

Finalmente, la reestructuración del proyecto conllevó un último grado de dificultad al tener que adaptar nuestras consultas con las definidas en el protocolo piro y el protocolo HTTP-

## **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 de países y cantones de Costa Rica, y mostrarlos al usuario desde un browser o desde la terminal.

## **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.
- Enviar una respuesta desde el servidor al cliente del país o cantón que solicite en un browser o terminal.

## 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).

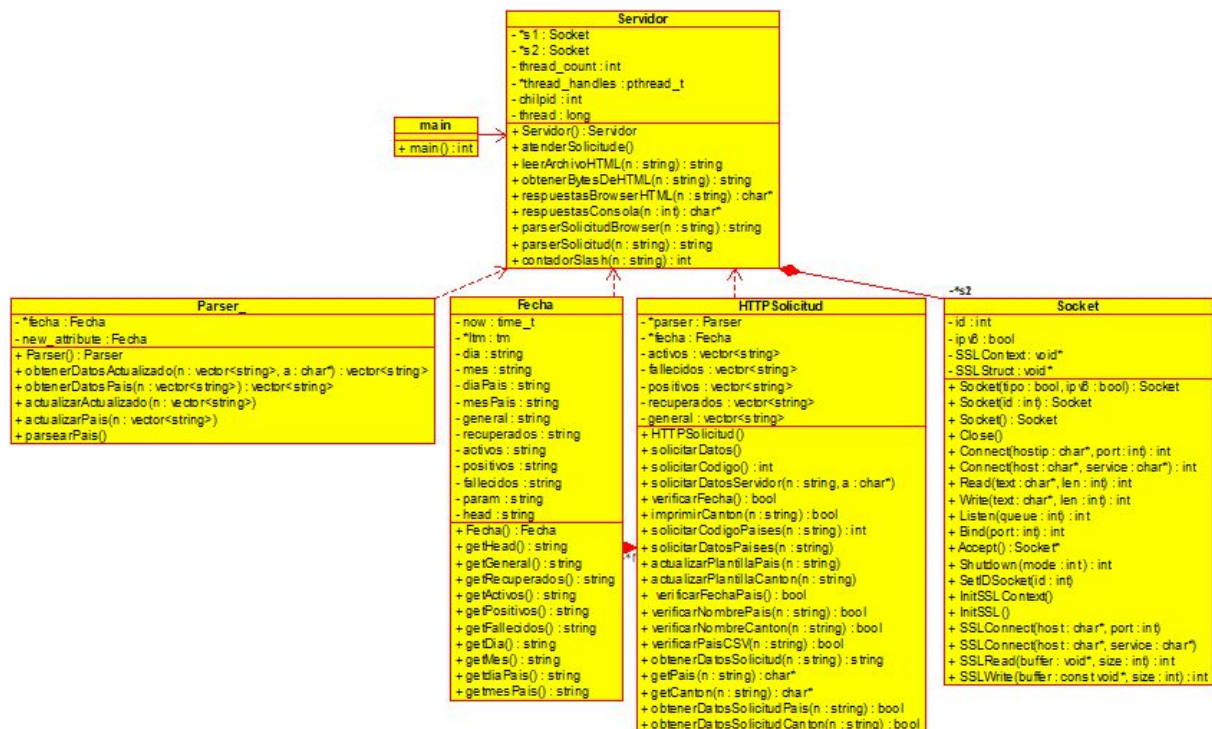
Una vez obtenida la respuesta del servidor, se procedió a parsear los datos y almacenarlos en un archivo .csv ( el archivo general se guardó en un documento llamado Pais.csv, mientras que los demás datos fueron guardados todos juntos en un solo documento llamado Actualizado.csv).

Lo anterior fue para el caso de Costa Rica, para el caso de países en general se utilizó igualmente una conexión utilizando socket al servidor del profesor Francisco Arroyo y de ahí obtenemos el país que solicite el cliente. El servidor que se creó se conecta con el cliente en el puerto 8080, este servidor recibe una solicitud del cliente que puede ser desde un browser o una terminal, en el caso de ser una solicitud de browser entonces se hace un parse de la primera línea en donde se solicita el recurso y si se encuentra lo pedido por el cliente entonces se escribe desde el servidor hacia el cliente una plantilla html que desplegará una tabla en el navegador con la información. Si se hace la solicitud desde consola, entonces primero en la consola se desplegará un menú donde el cliente elige si quiere ver la información de un país o un cantón de Costa Rica, luego ingresa el nombre y ese nombre se escribirá al servidor, luego el servidor determinará si se encuentra la información y la envía de vuelta al cliente, desplegando la información de manera tabulada. Es importante destacar que el servidor utiliza métodos de la clase que maneja las solicitudes HTTP porque es importante tener la información más reciente.

Manejo de Errores: en el caso de solicitudes a browser, se manejan los códigos 200, 400, 404, 501. El 200 corresponde a una solicitud válida entonces se desplegará la información que el cliente haya pedido. En el caso de que el cliente haga una solicitud inválida, como no pedir ningún recurso entonces se envía el mensaje 400 de petición inválida. Si el cliente pide algo y no se encuentra entonces se envía el 404 de información no encontrada. El 501 se implementó en caso de que se haga una solicitud de HTTP como DELETE o POST, ya que estas no se implementaron.

## 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	<p>Esta clase define los métodos y los atributos correspondientes a la clase Socket.cc.</p> <p>Librerías: -OpenSSL: Esta librería permite el uso de herramientas para el manejo de protocolos SSL/TLS.</p>
Socket.cc	<p>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.</p>
HTTPSolicitud.h	<p>Esta clase define los métodos y los atributos correspondientes a la clase HTTPSolicitud.cc.</p> <p>Librerías: -Fstream, iostream: Son librerías que permiten herramientas para el manejo de I/O dentro de cualquier programa.</p>
HTTPSolicitud.cc	<p>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.</p>
MainCliente.cc	<p>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.</p> <p>Librerías: -Fstream, iostream: Son librerías que permiten herramientas para el manejo de I/O dentro de cualquier programa.</p> <p>-Socket.h, HTTPRequest.h: Incluye ambas clases para el manejo del programa.</p>
Fecha.h	<p>Esta clase define los métodos y los atributos correspondientes a la clase Fecha.cc.</p> <p>Librerías: -Fstream, iostream: Son librerías que permiten herramientas para el manejo de I/O dentro de cualquier programa.</p>

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.

## 7. Limitaciones de su Solución

A pesar de los intentos, no se logró implementar el control del código de error 505, esto se debió a que, estaba generando problemas en la compilación al interactuar con las otras excepciones que, por cuestiones de manejo de tiempo, no se lograron corregir.

El manejo de países con guiones en sus guiones generaban un fallo al buscarlas, pues justamente se utilizó el guión medio (-) como delimitador para los

espacios en las palabras ingresadas en cada consulta, por lo que esta eventualidad no fue observada hasta el momento de las pruebas.

En algunas ocasiones, después de cierta cantidad de solicitudes, el servidor se puede caer, por lo cual se debe volver a ejecutar el comando `./servidor` y se puede seguir haciendo consultas con normalidad.

## 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.

### Pruebas con el Terminal

#### ***Caso#1: Muestra los datos de un cantón solicitado.***

```
Elija alguna de las opciones del menu
1. Consultar país
2. Consultar cantón
3. Salir del programa
2
Por favor digite el cantón a consultar
San-Ramon
Código  Provincia      Código  Cantón      Positivos  Activos  Recuperados  Fallecidos
2        Alajuela        202      San Ramón    865        378      480          7

Elija alguna de las opciones del menu
1. Consultar país
2. Consultar cantón
3. Salir del programa
```

#### ***Caso#2: Muestra los datos generales de un país.***

```
ricardopricardo-VirtualBox:/media/sf_ShareUbuntu/GitLab/Piro/c10123_proyectointegradorredessistemasoperativos_grupo0/src$ ./cliente
Elija alguna de las opciones del menu
1. Consultar país
2. Consultar cantón
3. Salir del programa
1
Por favor digite el país a consultar
Costa-Rica
País      Total Casos  + Casos  Total Muertes  Muertes  Total Recup  Recuperados  Activos  Criticos  Casos/mill  Muertes/mill  Total tests  Tests/mill  Población
Costa Rica  86,053      +1,225   1,055          +15      52,327      +545        32,671    197       16,851     207          251,285     49,207     5,106,729

Elija alguna de las opciones del menu
1. Consultar país
2. Consultar cantón
3. Salir del programa
```



**Caso#3: Muestra manejo de error, recurso no encontrado.**

```
Elija alguna de las opciones del menu
1. Consultar país
2. Consultar cantón
3. Salir del programa
1
Por favor digite el país a consultar
asdfasdf

Error! Recurso no encontrado

Elija alguna de las opciones del menu
1. Consultar país
2. Consultar cantón
3. Salir del programa
```

**Caso#4: Muestra manejo de error, solicitud inválida.**

```
Elija alguna de las opciones del menu
1. Consultar país
2. Consultar cantón
3. Salir del programa
2
Por favor digite el cantón a consultar

Error! Solicitud inválida

Elija alguna de las opciones del menu
1. Consultar país
2. Consultar cantón
3. Salir del programa
3
ricardo@ricardo-VirtualBox: /media/sf_ShareUbuntu/GitLab/Piro/cl0123_proyectointegradorredessistemasoperativos_grupoC/src$
```

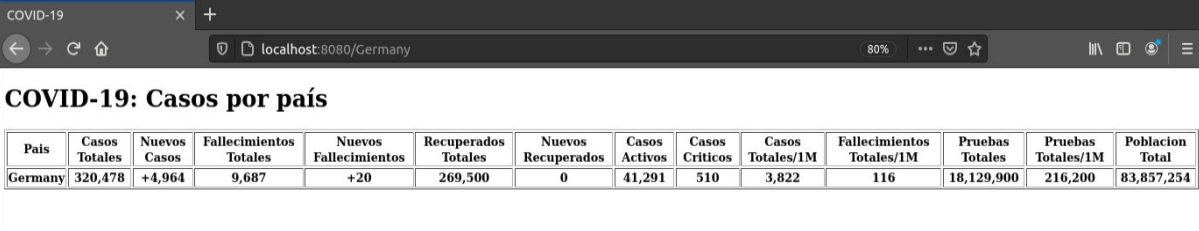
Pruebas con el Browser

**Caso#1: Muestra manejo de error, recurso no encontrado.**



The screenshot shows a web browser window with a single tab titled "COVID-19". The address bar displays "localhost:8080/Costa-Rica/asdas". Below the address bar, the page content shows the message "Error! Recurso no encontrado".

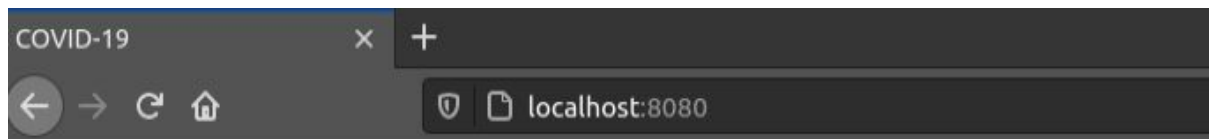
**Caso#2: Muestra los datos generales de un país.**



The screenshot shows a web browser window with the title 'COVID-19'. The address bar displays 'localhost:8080/Germany'. The page content is titled 'COVID-19: Casos por país'. Below the title is a table with 13 columns: País, Casos Totales, Nuevos Casos, Fallecimientos Totales, Nuevos Fallecimientos, Recuperados Totales, Nuevos Recuperados, Casos Activos, Casos Críticos, Casos Totales/1M, Fallecimientos Totales/1M, Pruebas Totales, Pruebas Totales/1M, and Poblacion Total. The table contains one row of data for Germany.


País	Casos Totales	Nuevos Casos	Fallecimientos Totales	Nuevos Fallecimientos	Recuperados Totales	Nuevos Recuperados	Casos Activos	Casos Críticos	Casos Totales/1M	Fallecimientos Totales/1M	Pruebas Totales	Pruebas Totales/1M	Poblacion Total
Germany	320,478	+4,964	9,687	+20	269,500	0	41,291	510	3,822	116	18,129,900	216,200	83,857,254

**Caso#3: Muestra manejo de error, solicitud inválida.**



**Error! Petición inválida**

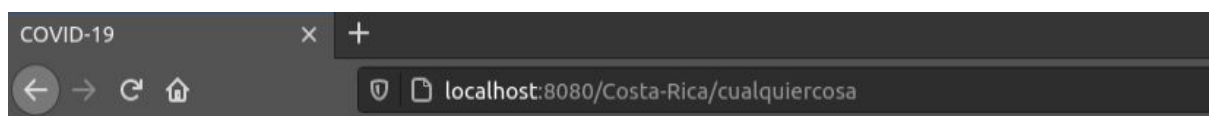
**Caso#4: Muestra los datos de un cantón solicitado**



The screenshot shows a web browser window with the title 'COVID-19'. The address bar displays 'localhost:8080/Costa-Rica/Leon-Cortes-Castro'. The page content is titled 'COVID-19: Casos por cantones'. Below the title is a table with 8 columns: Codigo Provincia, Provincia, Codigo Canton, Canton, Casos Positivos, Casos Activos, Casos Recuperados, and Fallecidos. The table contains one row of data for León Cortés Castro.

Codigo Provincia	Provincia	Codigo Canton	Canton	Casos Positivos	Casos Activos	Casos Recuperados	Fallecidos
1	San José	120	León Cortés Castro	41	6	35	0

**Caso#5: Muestra manejo de error, recurso no encontrado.**



**Error! Recurso no encontrado**

## 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:

### Cliente

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.

### Servidor Intermedio

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.

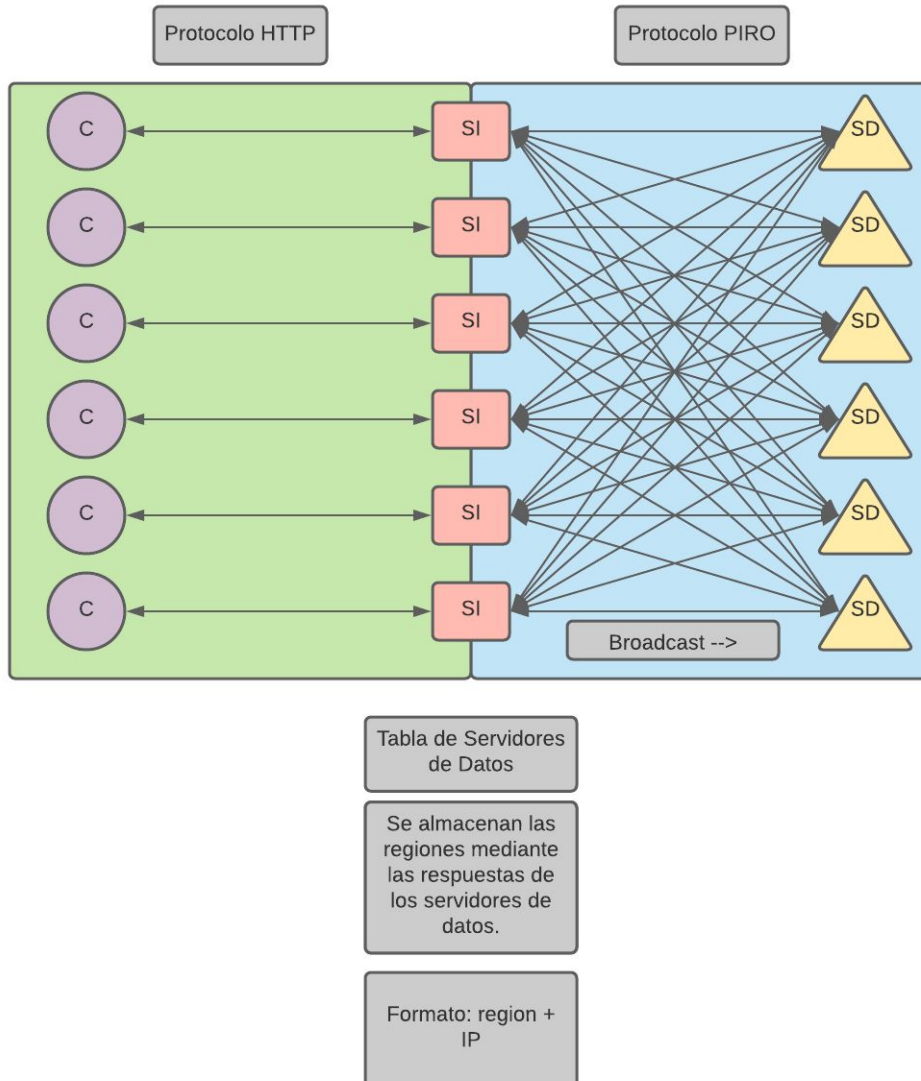
### Servidores de Datos

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:



## 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
9/22/2020	Definimos cómo corregir errores de la entrega anterior para adaptarlo a esta nueva entrega, hicimos una lista detallada de tareas por realizar
9/25/2020	Clase de PIRO Se hizo una reestructuración de clases y métodos para ir acorde con las buenas prácticas de programación.
9/26/2020	Se trabajó en el cliente nuevo, para poder hacer las consultas por cantón escribiendo el nombre en vez de utilizando un menú
9/27/2020	Conversamos sobre el protocolo, distribuimos tareas, comenzamos a trabajar en las modificaciones al servidor para que funcionara como servidor y cliente
9/29/2020	Se finalizó la funcionalidad básica del servidor, donde se podía hacer una búsqueda ingresando los datos en el buscador
10/2/2020	Se añadió código para manejar errores, la utilización correcta de las mayúsculas, pasos de mensaje, así como comunicación entre cliente de consola y servidor
10/3/2020	Se realizaron correcciones en el código, para un parseo más optimizado de los datos enviados tanto al cliente en consola como al cliente en el browser
10/6/2020	Se corrigieron errores que surgieron en el manejo de buffer y también errores inesperados que surgieron al hacer pruebas con nombres de países y cantones de más de dos palabras
10/7/2020	Se avanzó paralelamente en la documentación interna y externa del código
10/8/2020	Corrección menor de errores, eliminación de código basura, revisión de buenas prácticas
10/9/2020	Mejoras finales al código, ejecución de pruebas, finalización de documentación

## 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>.