

Laboratorio 6

Introducción

En este laboratorio de clientes y servicios, se trabajó primero qué todo con los métodos que ofrece la clase URL para obtener información como el protocolo o el puerto de una URL, como siguiente se vio la implementación de los sockets, en primer caso como uso de conexión entre servidor y cliente, en el cual el cliente enviaba un mensaje y el servidor respondió con el mismo mensaje pero con una respuesta diferente para notar que el servidor era el que respondía. Al ver el uso y aplicación que tienen los sockets se implementó el uso de estos sockets pero para un servidor web, el cual respondía a peticiones GET que se hacen desde un cliente, en este caso un browser, el cliente era capaz de responder a estas peticiones dependiendo del tipo de contenido que se iba a mostrar en el navegador. Al ver la aplicación de estos sockets en este caso desde un cliente como un browser, se vio la aplicación y los usos de los datagramas, los cuales son capaces de recibir mensajes y responderlos, con la diferencia que estos servidores son capaces de recibir mensajes y enviar mensajes, pero si el cliente no recibe un mensaje, este tiene la posibilidad de esperar un tiempo a que le llegue el mensaje, y si este no llega, cancelar la conexión con este. Después de ver los usos de los datagramas se vio el uso de los RMI, los cuales consisten en la conexión entre dos programas que están corriendo en diferentes máquinas virtuales de java, estos permiten que un programa pueda llamar remotamente los métodos que el otro programa tenga dispuestos para que sean llamados de esta manera, estos métodos pueden ser llamados simplemente con el nombre del método que está implementado en la interfaz remota.

Conceptos Generales

TCP, UDP, Servidor, Cliente, Socket, URL, Servidor web, Datagrama, Protocolo, Mensaje.

Implementación

Para la implementación de cada uno de estos ejercicios de este laboratorio se usó como referencia los códigos que estaban descritos en cada punto como ejemplos. Al tomar como referencia estos códigos, se tenía una gran ayuda para la implementación de los problemas que se debían resolver.

Para el primer ejercicio se usaron los métodos que ofrece el método URL para ver la información que contiene un objeto URL, después se debió hacer una implementación correcta para obtener toda la información html que contiene una página web dada, esta información guardarla en un archivo con extensión .html para que sea posible abrirlo en un browser el código que ha sido guardado.

Para el segundo ejercicio se debieron implementar los sockets de servidor y cliente, esto para que fuera posible la comunicación cliente-servidor, en los cuales un cliente enviaba una petición o un mensaje y el servidor responde con otro mensaje o con la resolución de la petición dada por el cliente. Para que el cliente y el servidor en el primer caso pudieran enviar y leer los mensajes que se enviaban entre ellos, era necesario crear instancias BufferedReader para la entrada y la salida de datos, en este caso de los mensajes, así sería posible la lectura y escritura de estos, para el segundo caso fue necesario editar los encabezados del protocolo HTTP/1.1 para que el navegador el cual en este caso funcionaba

como cliente le fuera posible mostrar las peticiones con los recursos que el servidor enviaba, en estos casos páginas texto o imágenes de extensión .png.


Para el tercer ejercicio se implementó algo parecido a el anterior punto de un servidor y un cliente, pero en este caso se usaron datagramas, los cuales permiten el envío y recepción de mensajes, pero con la diferencia que no importa si el mensaje no es enviado o recibido, o el tiempo de demora de envío de un mensaje, por lo tanto se crean instancias de objetos de Datagrama para estos mensajes.

Para el último ejercicio se usaron métodos remotos para dos programas que se ejecutan en diferentes máquinas virtuales de java, esto para permitir que los dos programas interactúan entre sí con el llamado de métodos entre sí, esto se lograba con la implementación de interfaces remotas, y luego implementando los métodos que fueron puestos en las interfaces para que los otros programas les fuera posible llamarlos o referenciarlos.

Pruebas

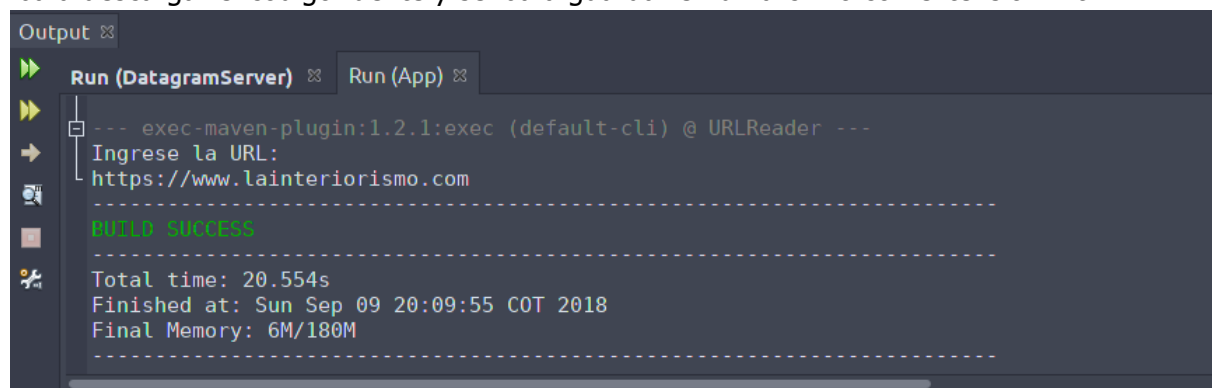
Ejercicio 1

Se puede observar el llamado que se pide en el ejercicio de los métodos que ofrece el objeto URL.



```
Output
Run (DatagramServer) Run (App)
--- exec-maven-plugin:1.2.1:exec (default-cli) @ URLReader ---
getProtocol() http
getAuthority() www.lainteriorismo.com:8080
getHost() www.lainteriorismo.com
getPort() 8080
getPath() /docs/
getQuery() a=b
getFile() /docs/?a=b
getRef() Downloading
-----
BUILD SUCCESS
```

En este punto se pedía una URL la cual debía de ser tipo <https://xxxxxxx.xx>, de la cual se iba a descargar el código fuente y se iba a guardar en un archivo con extensión .html



```
Output
Run (DatagramServer) Run (App)
--- exec-maven-plugin:1.2.1:exec (default-cli) @ URLReader ---
Ingrese la URL:
https://www.lainteriorismo.com
-----
BUILD SUCCESS
-----
Total time: 20.554s
Finished at: Sun Sep 09 20:09:55 COT 2018
Final Memory: 6M/180M
-----
```

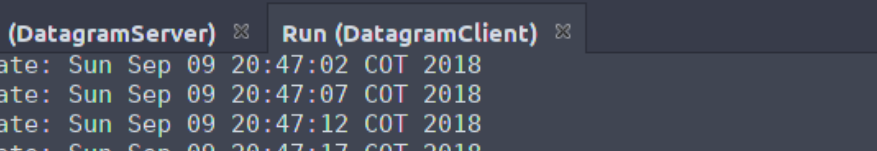
```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta http-equiv="X-UA-Compatible" content="IE=Edge"/>
5
6   <meta charset="utf-8"/>
7   <meta name="generator" content="Wix.com Website Builder"/>
8   <link rel="shortcut icon" href="https://static.wixstatic.com/media/1842fb_57fd3c1f40624d78874eb451fb5154c7%7Emv2.png/v1/fill/w_32%2Ch_32%2Clg_1%3A1%3A1/skype.ico?w=32&h=32&format=webp&compress=lossy">
9   <link rel="apple-touch-icon" href="https://static.wixstatic.com/media/1842fb_57fd3c1f40624d78874eb451fb5154c7%7Emv2.png/v1/fill/w_32%2Ch_32%2Clg_1%3A1%3A1/skype.png?w=32&h=32&format=webp&compress=lossy">
10
11
12
13
14   <meta http-equiv="X-Wix-Meta-Site-Id" content="ab24dbab-c577-4c46-b931-7b1bd54c52c7"/>
15   <meta http-equiv="X-Wix-Application-Instance-Id" content="2a3fd1c1-2ebc-45d4-9c1d-c334ce08a569"/>
16
17   <meta http-equiv="X-Wix-Published-Version" content="326"/>
18
19   <meta http-equiv="etag" content="17eac9ce9c91bc3dbbf36c78ff34a733"/>
20
21
22
23
24
25   <meta name = "format-detection" content = "telephone=no"/>
26
27
28
29
30   <meta name="SKYPE_TOOLBAR" content="SKYPE_TOOLBAR_PARSER_COMPATIBLE"/>
31

```

En este ejercicio tocaba implementar un servidor el cual al recibir un mensaje en este caso un número, éste respondiera con el cuadrado del mismo.

Ejercicio 3

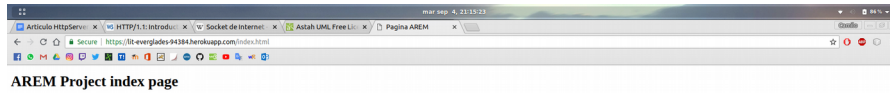


The screenshot shows a Java IDE with two tabs: "Run (DatagramServer)" and "Run (DatagramClient)". The "Run (DatagramServer)" tab is active, displaying the following output:

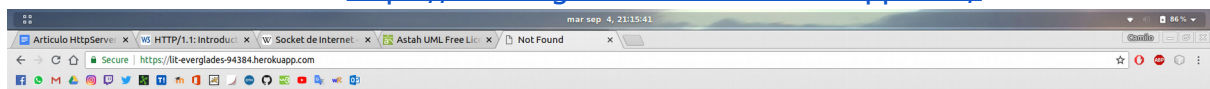
```
Date: Sun Sep 09 20:47:02 COT 2018
Date: Sun Sep 09 20:47:07 COT 2018
Date: Sun Sep 09 20:47:12 COT 2018
Date: Sun Sep 09 20:47:17 COT 2018
Date: Sun Sep 09 20:47:22 COT 2018
Date: Sun Sep 09 20:47:27 COT 2018
Date: Sun Sep 09 20:47:32 COT 2018
Date: Sun Sep 09 20:47:37 COT 2018
Date: Sun Sep 09 20:47:42 COT 2018
Date: Sun Sep 09 20:47:47 COT 2018
```

Ejercicio 4

En esta imagen se puede observar la aplicación desplegada en Heroku, en este caso abriendo el html con nombre index.html y el link en heroku: <https://secret-crag-85339.herokuapp.com/UrlWeb.html> se observa que la página despliega correctamente y que la petición al archivo se hace correctamente.

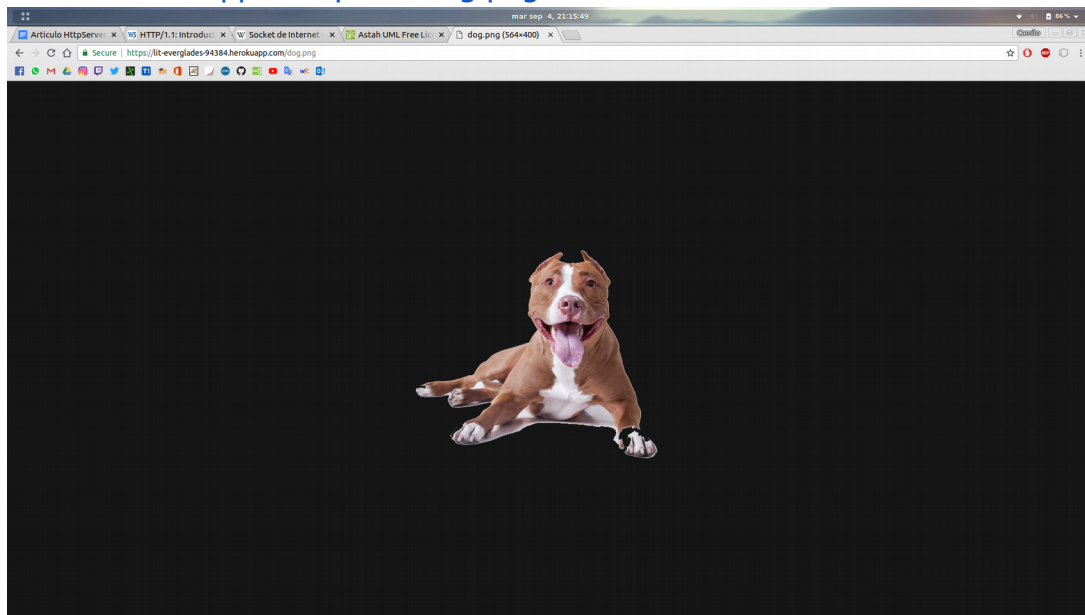


En esta imagen se puede observar que al hacer una petición a un archivo o página no contemplado en el proyecto, se despliega en mensaje Page not found, haciendo saber al usuario que la petición que está realizando no está disponible. Esta petición se realiza desde el link: <https://lit-everglades-94384.herokuapp.com/>



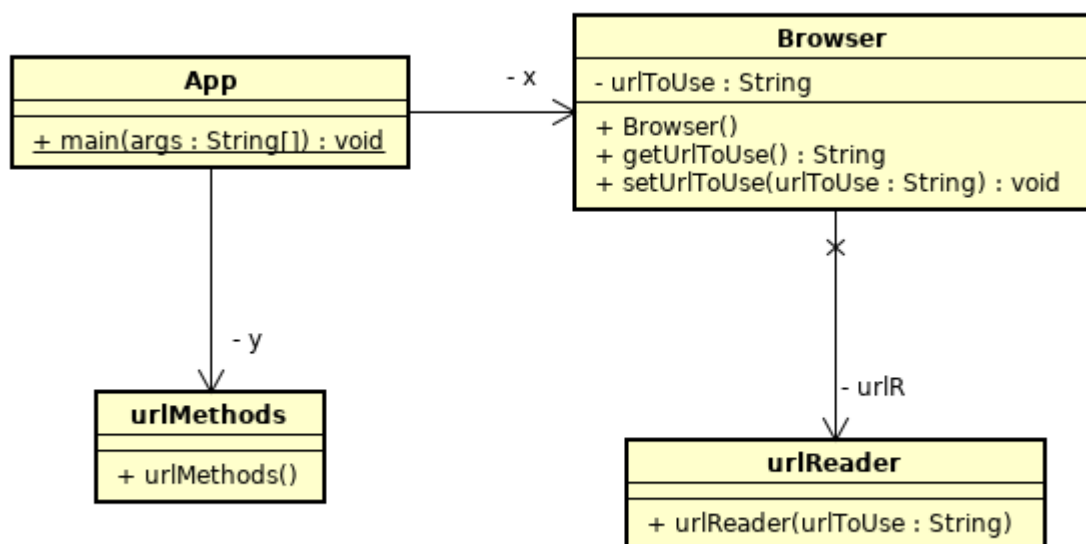
Page not found

En esta imagen se puede observar qué al hacer la petición get de la imagen tipo png se muestra con fondo negro ya que esta imagen se supone no tiene fondo, está petición se realiza con el link: <https://secret-crag-85339.herokuapp.com/pitbullDog.png>

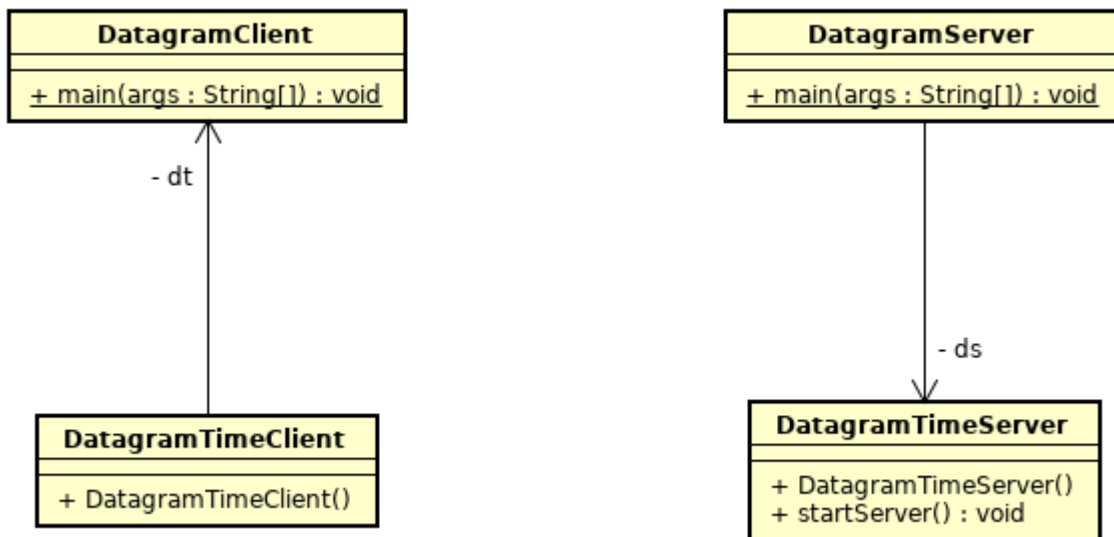


Gráficas

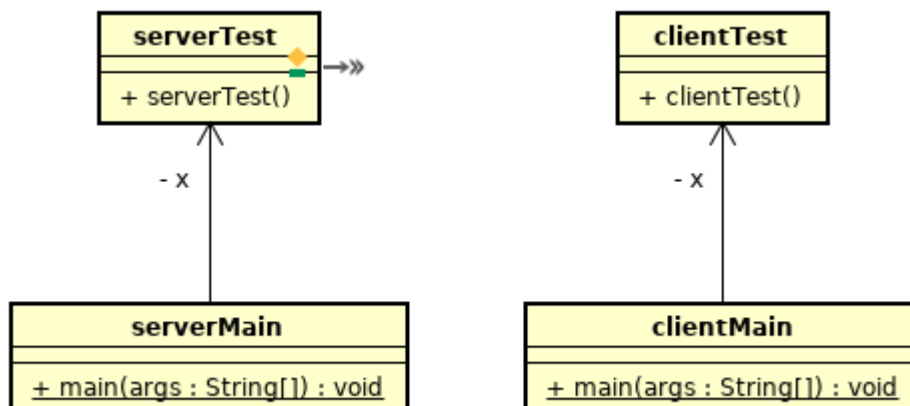
Para el primer ejercicio se usó un main para llamar las clases urlMethods() y Browser(), la cual invoca la clase urlReader, la clase App invoca urlMethods con la instancia y Browser con la instancia x.



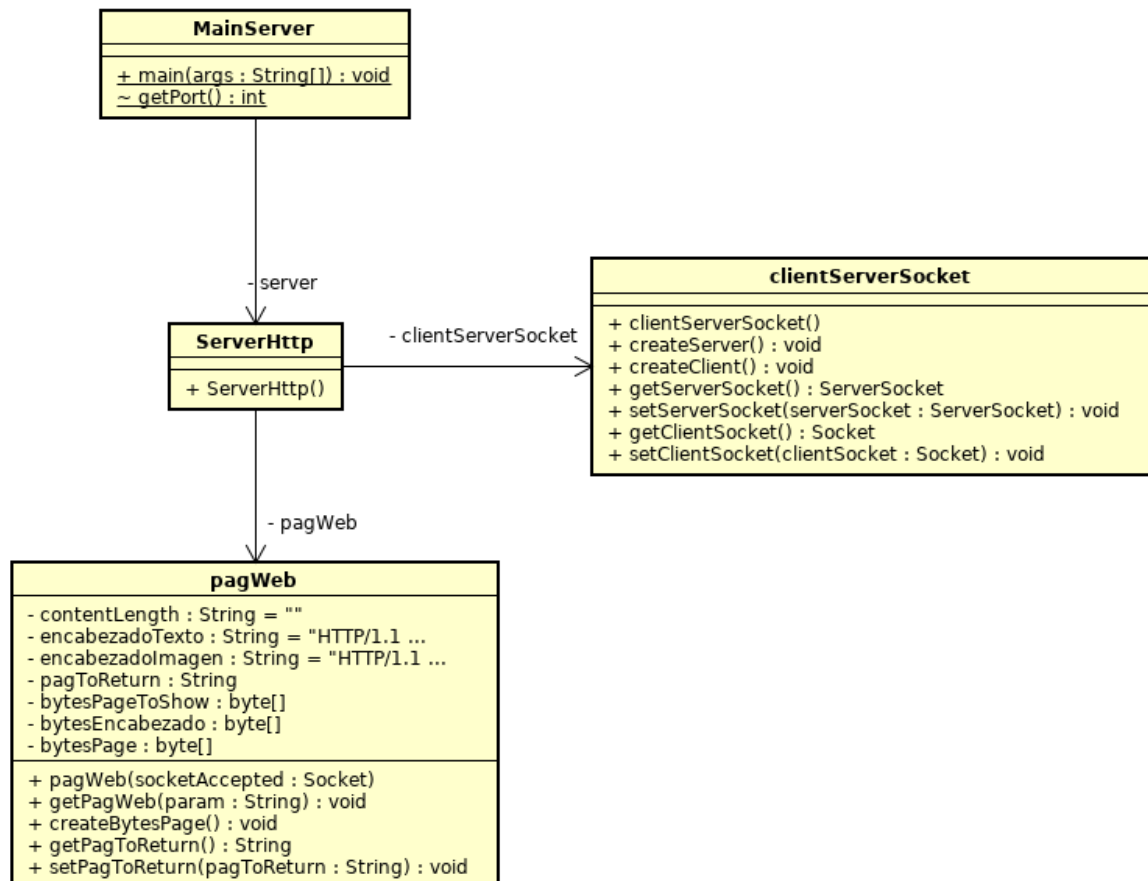
Para el ejercicio de los datagramas no más se tienen 2 clases importantes las cuales son el cliente y el servidor, las otras dos clases se encargaban de invocar el DatagramClient y el DatagramServer con las invocaciones dt y ds respectivamente.



Para el punto en el que el servidor responde al mensaje de un cliente con el cuadrado de un número, se utilizaron 4 clases, dos importantes y dos las cuales se encargan de invocar el servidor y el cliente, `serverTest` y `clientTest` son invocadas por las clases `serverMain` y `clientMain` respectivamente con las instancias `x` y `x`.



Se implementan 4 clases, de las cuales una está conectada con todas qué en este caso es la de `HttpServer`, la cual es la qué ejecuta el `main`, la qué crea los sockets de servidor y cliente y la qué además hace qué la página se despliegue con su propio encabezado pero todo esto convertido a `byte[]`. La clase `main` se encarga de crear el `HttpServer` y además dar el puerto por defecto qué usará heroku para desplegar la aplicación, en este caso el puerto número 35000.



Conclusiones

En éste laboratorio pudimos observar el uso de los servidores y los clientes, como se puede conectar mediante los protocolos UDP y TCP, los cuales para estos casos sirven para el envío y la recepción de mensajes cliente-servidor, además vimos qué el cliente no necesariamente tiene que estar corriendo en el JVM, sí no qué como se vio en el servidor web puede ser un navegador, el cual también es capaz de mostrar los mensajes que envía el servidor en respuesta a las peticiones que realiza el cliente, además de estos servidores, pudimos observar que existen servidores y clientes los cuales pueden funcionar juntos sin que necesariamente los mensajes entre ellos lleguen o se demoren un tiempo prudente en llegar, ya que estos mensajes no afectan los procesos de funcionamiento de cliente-servidor. De las clases remotas, podemos observar que tienen un uso muy importante, ya que dos programas pueden estar funcionando en diferentes JVM, pero sí tienen una buena implementación de los métodos que son creados en las interfaces y además con una buena localización y referencia de estos métodos, el funcionamiento de estas clases remotas tiene usos muy importantes para las necesidades actuales de la tecnología.