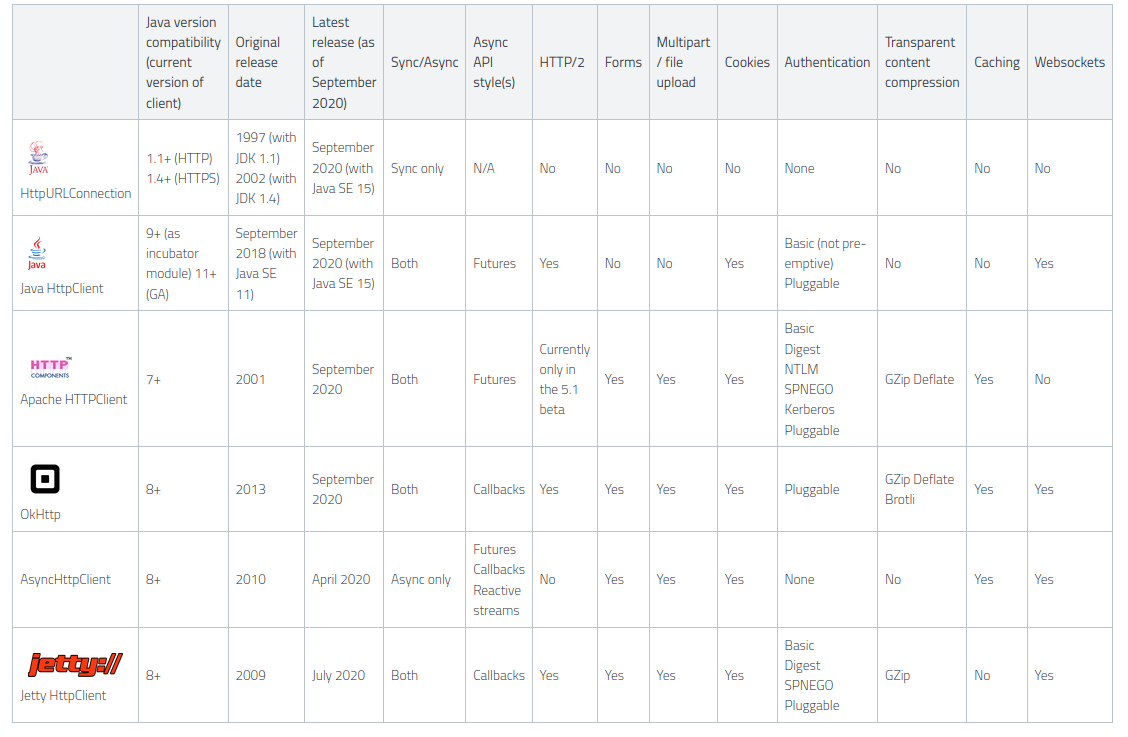
Para la implementación del proxy HTTP, se decidió llevar a cabo un proceso de multitarea, esto principalmente para evitar demoras y problemas a la hora de hacer uso del proxy, puesto que si bien, la computación moderna es muy eficiente y por ende las tareas se ejecutan rápidamente, siempre existe la posibilidad que el proxy no este disponible en el momento en el que se intenta hacer una consulta web, lo que llevaría en el peor de los casos a que deba de hacerse esta petición otra vez. Es importante mencionar aquí que existen dos formas de llevar a cabo un proceso de multitarea: Multitarea basada en hilos, que consiste básicamente en separar un espacio en el proceso que está en ejecución actualmente para ejecutar otra tarea, y multitarea basada en procesos, que consiste en crear un proceso nuevo cada vez que se quiere llevar a cabo una tarea diferente.

El acercamiento que se escogió para esta implementación fue un acercamiento de un proceso de multitarea basado en hilos, lo que permite que simultaneamente se ejecuten diversas tareas en un mismo proceso. Este acercamiento permite que los tiempos de ejecución de un proceso específico se puedan minimizar, puesto que se busca dividir el problema en problemas más pequeños y que cada pequeño problema se pueda solucionar de forma simultánea en una unidad de ejecución diferente, en este caso hilos. Esto presenta una ventaja importante en cuanto al tiempo de ejecución del proceso en comparación con un modelo completamente secuencial, es decir que ejcuta una serie de instrucciones todas en un mismo hilo, seguidas una de otra. La razón por la que se decidió seguir un proceso de multitarea basado en hilos y no en procesos resulta en que no solo demora un tiempo relativamente menor crear un hilo dentro de un proceso ya existente en comparación con crear un proceso completamente nuevo, pero también se comparten recursos y se disminuye en parte el esfuerzo requerido para entregar al cliente el resultado de su solicitud.

Una vez definida la forma en la que se va a llevar a cabo la distribución de las diferentes solicitudes que recibe el proxy, es importante definir la forma en la que se realizará la petición al servidor HTTP que guarda el recurso solicitado. Para esto, se investigó acerca de las diferentes opciones que existen para cumplir con esta tarea. Tras investigar que existen bastantes implementaciones especialmente en el lenguaje JAVA, se buscó alguna comparación que existiera entre las principales opciones. De esta forma, llegamos a la siguiente tabla comparativa, en la que se muestran algunas de las características más importantes de un cliente http, comparadas entre las principales opciones existentes para realizar peticiones HTTP.

Imagen tomada de: https://www.mocklab.io/blog/which-java-http-client-should-i-use-in-2020/

Adicional a estas opciones, también se investigó un poco sobre el uso de Sockets TCP para cumplir con esta labor, sin embargo se llegó a la conclusión que el uso de Sockets TCP no es del todo recomendado puesto que los sockets pertenecen a la capa de sesión o capa 5 en el modelo OSI, mientras que el protocolo HTTP pertenece a la capa de aplicación o capa 7 en el modelo OSI, esto significa que sería necesario implementar desde cero el protocolo HTTP y por consiguiente resulta más eficiente hacer uso de alguna de las clases referenciadas en la tabla.

Tras un análisis de la información propia de la tabla de comparación, fue posible identificar que para el caso de un proxy http que haga uso de la version del protocolo 1.0, la clase más pertinente resulta siendo la proporcionada por la fundación Apache, dado que esta es sencilla de entender, tiene una amplia gama de funcionalidades, recibe soporte activamente por parte de la fundación Apache y también es posible acomodar su funcionamiento para cumplir con las necesidades que se vean planteadas en un momento específico. Estas características convierten a este cliente HTTP en una opción interesante para la implementación del proxy HTTP.

<https://www.javatpoint.com/java-http-proxy-server>

<https://www.frlp.utn.edu.ar/materias/internetworking/apuntes/http/httpeasy.pdf>

<https://github.com/andrii-matviichuk/java-http-proxy/>

<https://www.infor.uva.es/~fdiaz/sd/doc/hilos>

<https://www.cs.buap.mx/~mtovar/doc/PCP/EjemplosHilos.pdf>

<https://javadesdecero.es/avanzado/programacion-multihilo/>

<https://systope.blogspot.com/2012/05/procesos-e-hilos.html>

https://stackoverflow.com/questions/1611931/catching-ctrlc-in-java

https://www.codejava.net/java-se/networking/how-to-use-java-urlconnection-and-httpurlconnection

https://www.w3.org/Protocols/HTTP/1.0/spec.html

https://condor.depaul.edu/dmumaugh/readings/handouts/SE435/HTTP/node11.html

https://www.journaldev.com/7148/java-httpurlconnection-example-java-http-request-get-post

https://mkyong.com/Java/how-to-send-http-request-getpost-in-Java/

[https://books.google.com.co/books?hl=es&lr=lang\_es&id=ry8bAgAAQBAJ&oi=fnd&pg=PP1&dq=protocolo+http+java&ots=Q7bkzvy4bE&sig=8\_7GW7V9\_mt-SjHBdaxQKsO43s0&redir\_esc=y#v=onepage&q=protocolo%20http%20java&f=false](https://books.google.com.co/books?hl=es&lr=lang_es&id=ry8bAgAAQBAJ&oi=fnd&pg=PP1&dq=protocolo+http+java&ots=Q7bkzvy4bE&sig=8_7GW7V9_mt-SjHBdaxQKsO43s0&redir_esc=y" \l "v=onepage&q=protocolo http java&f=false)