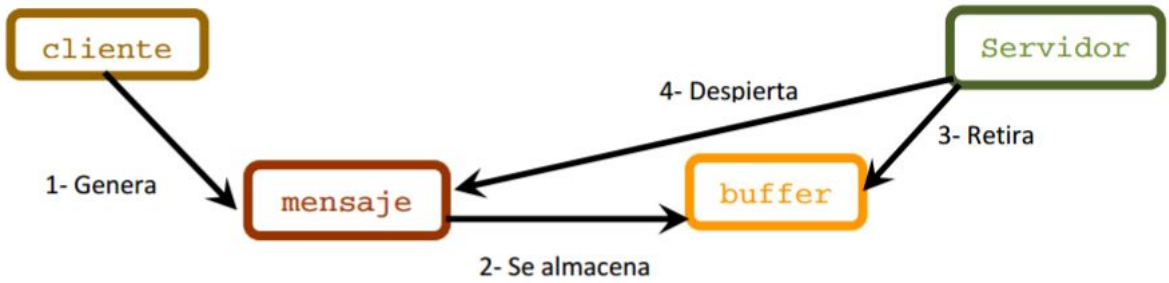
Juan Sebastián Gómez - 201414022
Andrés Felipe Moreno - 201214229
Juan Guillermo Murillo - 201423979

Documentación Caso 1

Diseño

Para la solución de este caso, se decidió emplear el diseño presentado en el enunciado, el cual se presenta a continuación:



Observando el diseño, decimos que se está planteado un diseño Productor-Buffer-Consumidor. Este diseño utilizará como consumidor a la clase servidor, pues estará en constante consulta de la clase buffer para extraer y procesar los mensajes que allí se encuentren almacenados. El papel de productor lo tendrá el cliente, quien generará los mensajes que desea enviar al servidor; sin embargo, se realiza una variación de la comunicación productor-buffer, pues será el mensaje generado por el cliente quien se comunique directamente con el buffer.

De manera global, el sistema funciona de manera concurrente, recibiendo una cierta cantidad de mensajes por cada cliente. De estos mensajes generados por los clientes se almacenará una cantidad determinada de ellos en el buffer. Aquellos mensajes que no puedan ser almacenados en el buffer harán que el mensaje quede dormido en el buffer hasta que sean notificados para almacenar el mensaje (espera pasiva). Una vez se almacene el mensaje en el buffer para que sean procesados por los servidores, el cliente quedará a la espera de la respuesta del servidor pero quedará dormido sobre el mensaje (espera pasiva). Luego de que se almacenen los mensajes, cada servidor intentará retirar mensajes que se encuentren alojados en el buffer, si el buffer se encuentra vacío, el servidor entrará en una espera activa por el próximo mensaje que se almacene en el buffer. En cada intento el servidor cede el procesador para que otro servidor pueda usarlo. Una vez sea retirado el mensaje, el servidor debe generar una respuesta la cual va a despertar el mensaje que se encontraba en espera pasiva con lo cual también se despertará al cliente asociado al mensaje. Una vez sean procesados los mensajes de un cliente, este debe reportar al buffer, a través de un mensaje, que ya se retirará de los clientes activos.

Juan Sebastián Gómez - 201414022
Andrés Felipe Moreno - 201214229
Juan Guillermo Murillo - 201423979

Sincronización

La sincronización de la solución inicia con con la ejecución del método `send()` del mensaje, pues será aquí donde, una vez alojado en el buffer, este último debe esperar a que un servidor se encuentre disponible para procesar el mensaje anteriormente almacenado; es decir que se realiza una espera pasiva o un llamado al método `wait()` en el buffer. Esta espera se mantendrá hasta que un servidor disponible retire el mensaje, lo procese para modificarlo y lo despierte para que el mensaje pueda ser retornado al cliente y este último pueda verificar si su mensaje ya fue procesado o no. Adicionalmente, hay sincronización cuando el buffer se encuentra lleno, lo que ocasiona que los nuevos mensajes que deseen ingresar al buffer serán puestos en una espera pasiva, se invoca al método `wait()` del mensaje, hasta que el buffer tenga espacio disponible para aceptar nuevos mensajes.

Interacciones entre objetos

A partir de este punto se comenzará a hablar de manera particular con respecto a las diferentes parejas de objetos que se encuentran en la solución desarrollada.

- **Cliente - Mensaje**

  Como se plantea en el diseño de la solución, el cliente solamente se comunica con los mensajes que son de su autoría. El cliente posee un arreglo de mensajes los cuales serán enviados al buffer para su procesamiento. Dentro del método `run()` del cliente se genera el contenido de los mensajes y se invoca al método `send()` de los mensajes para que estos sean enviados al buffer, donde los mensajes encolados se dormirán hasta que llegue un servidor y los despierte para su posterior procesamiento. Cada mensaje se tiene por atributos el contenido del mensaje y una conexión con el buffer.

- **Mensaje - Buffer**

  Una vez el mensaje ha sido enviado al buffer, este entrará en una espera pasiva hasta que sea procesado y despertado por el servidor. A partir del momento en que el mensaje se duerme, el buffer almacena el mensaje si aun tiene espacio de almacenamiento; de lo contrario entrará en una espera pasiva hasta que algún mensaje sea extraído y permita el ingreso del mensaje que fue enviado. El buffer contiene diferentes atributos y métodos encargados de realizar las verificaciones pertinentes para decidir si puede almacenar mensajes o si debe encolarlos para almacenarlos posteriormente.

- **Servidor - Buffer**

  La interacción entre el servidor y el buffer es la interacción más sencilla de todas pues el servidor, a través de su método `run()` le pregunta al buffer si tiene mensajes. En caso que la respuesta sea afirmativa el servidor procede a extraer un mensaje del buffer, de lo contrario el servidor cederá el procesador a través del método `yield()`

y preguntará nuevamente por un mensaje cuando tenga la oportunidad de hacerlo. La clase servidor es a su vez la clase más importante de la solución, pues es esta quien posee el método `main()` se crea el escenario de prueba del problema (se cargan las 'properties'),

- **Servidor - Mensaje**
  Una vez el servidor extrae un mensaje, este pasa a ser procesado por medio del método `pop()` del buffer, el cual despierta al mensaje retirado del buffer, procede a modificarlo por medio del método `modify()` de la clase mensaje para editar el contenido del mensaje. Luego de que el mensaje es modificado y, gracias al método mencionado anteriormente, el mensaje realiza la notificación al cliente para que realice las verificaciones pertinentes sobre el procesamiento del mensaje.