

**Infraestructura computacional**

**Ingeniería de Sistemas y Computación**

**ISIS-2203**

Daniela Mariño - 201412576

Joan Torres - 201315711

María Arévalo - 201415326

**Caso 1 : Logística y Seguridad Aeroportuaria S.A**

**Diseño y funcionamiento**

Para el caso de estudio “Logística y Seguridad Aeroportuaria S.A” en el que se requería manejar consultas de varios clientes sobre un servidor se tomó la decisión de crear 5 clases, Admin, Buffer, Cliente, Mensaje y Servidor. La clase Admin es la encargada de procesar el archivo properties con el número de clientes, el número de servidores y el tamaño del buffer en el main del programa. Con estos datos, esta clase inicializa el buffer y la conexión. La clase Buffer inicializa los clientes y servidores además de crear una lista para guardar los mensajes entrantes y los clientes en cola.

La clase Cliente, al ser inicializada, genera un objeto de tipo Mensaje. Para generar un mensaje, en este se define el id del mismo y su remitente. Una vez creado el mensaje, este es enviado por el cliente al buffer, el cual sincroniza la lista de mensajes, y si el número de mensajes enviados es menor al tamaño del buffer, lo agrega a la lista de mensajes. Si el cliente no puede enviar el mensaje al buffer, este se queda en espera activa, pero debe ceder el procesador, esto se hace mediante el método yield. Cuando el mensaje se haya depositado, el cliente queda a la espera siendo esta pasiva, durmiendo sobre el objeto mensaje.

El servidor es el encargado de retirar los mensajes, depositados por el cliente, del buffer. Para esto, el servidor tiene un mensaje actual, en cuyo caso este sea igual a null, va a solicitar un nuevo mensaje, llamando al método pedirMensaje() en el buffer. Este método sincroniza la lista de mensajes, y si esta es mayor a 0, devolverá el mensaje en la primera posición al servidor, y a su vez lo retirará de la lista de mensajes del buffer. Así como para el cliente, si el servidor no pudo retirar el mensaje del buffer, este se queda en espera activa, cediendo el procesador mediante el método yield. Después de ser obtenido el mensaje, se envía la respuesta a este mediante el método responderMensaje(), que llama al método recibirRespuesta() de mensaje, enviando una respuesta por defecto “OK”. El objeto mensaje la guarda y despierta al cliente que remitió el mensaje, notificando que ya tiene respuesta.

Finalmente, el thread que simula el cliente se despierta y activa el metodo recibirRespuesta(), en donde suma +1 al número de mensajes respondidos y además si este número es igual al número de mensajes a enviar, esto significa que ya terminó, por lo que con el método terminar() le avisa al buffer, para que éste lo retire de la lista de clientes y reduzca el número actual de clientes.