El **gestor de colas** proporcionará las herramientas necesarias para almacenar y gestionar todos los datos del resto de procesos.

Implementación:
Primero, es necesario almacenar los datos que el resto de procesos envían. Para ello, se utiliza la cola genérica *BoundedQueue* ya dada.
Para evitar el solapamiento de entrada y salida de los datos en las colas, se usa un monitor cuyo esquema en alto nivel es el siguiente:

```
Monitor ControldeCola
        integer numElementos := 0
        boolean fin := false
        condition estaEscribiendo
        condition estaBorrando

operation escribirCola( BoundedQueue cola, string elemento )
        while (numElementos >= MAX_ELEMENTOS)
                waitC( estaEscribiendo )
        end
        cola.encolar(elemento)
        numElementos := numElementos + 1
        signalC (estaBorrando)
end

operation leerCola(BoundedQueue cola, string elemento)
        boolean leido = false
        while (numElementos <= 0 && !fin)
                WaitC(estaBorrando)
        end
        if (numElementos > 0)
                numElementos - -
                cola.desencolar(elemento)
                signalC(estaEscribiendo)
                leido= true;
        end
        return leido
end


operation finalizar()
        fin := true
        signalC_all( estaBorrando )
end
```

El monitor está diseñado de tal forma que para cada cola que se necesite, se invoque un monitor.
El monitor cuenta con dos variables condición y dos operaciones. Si se quiere escribir en la cola primero se comprueba que la cola no esté llena y si no lo está se añade el elemento deseado al final de la cola. Para leer un elemento de la cola se comprueba primero que la

cola no es vacía y si no lo es se guarda el primer elemento y después se desencola. Si fuera vacía, se devuelve falso en un booleano de control.

El programa principal crea dos sockets: uno para el master y los workers y otro para los analizadores. A continuación lanza N + 1 procesos master/worker (siendo N el número máximo de procesos que se pueden conectar) y dos procesos analizadores. A continuación se detalla el funcionamiento de ambos tipos de proceso.

```
Channel of string socTareas, socAnalizadores
ControldeCola controlTareas, controlTags, controlQoS
BoundedQueue colaTareas, colaTags, colaQoS

Process masterWorker
        string mensaje
        bool out := false
        while (!out)
                socTareas => mensaje
                if (datos = “FIN”)
                        out := true
                else if (mensaje = “PUBLISH_TAREAS,datos”)
                        controlTareas.escribirCola(colaTareas, datos)
                else if (mensaje =  “READ_TAREAS”)
                        if (controlTareas.leerCola(colaTareas, datos) )
                                socTareas <= datos
                        else
                                mensaje := “FIN”
                                socTareas <= mensaje
                        end
                else if (mensaje = “PUBLISH_QoS”)
                        controlQoS.escribirCola(colaQoS, datos)
                else if (mensaje == “PUBLISH_TAGS”)
                        controlTags.escribirCola(colaTags, datos)
                end
        end
end
```

El proceso *masterWorker* es el encargado de gestionar la comunicación con los procesos master/worker dependiendo de lo que escuche por su canal. Por ejemplo si un proceso worker quiere leer tareas pendientes, se desencolara de la cola de tareas los datos correspondientes y se le enviaran estos. Si no quedaran tareas pendientes se le envia un mensaje de fin y se finaliza la comunicación.

```
Process analizadores
        string mensaje
        boolean out := false
        while(!out)
```

```
                socAnalizadores => mensaje
                if (mensaje = “READ_QoS”)
                        if (controlQoS.leerCola(colaQoS, mensaje))
                                socAnalizadores <= mensaje
                        else
                                mensaje := “FIN”
                                socAnalizadores <= mensaje
                                out := true
                else if (mensaje = “READ_TAGS”)
                        if (controlTags.leerCola(colaTags, mensaje)
                                socAnalizadores <= mensaje
                        else
                                mensaje := “FIN”
                                out := true
                        end
                end
        end
end
```

El proceso *analizadores* se encarga de recibir peticiones de lectura por parte de los analizadores y enviar los datos correspondientes. Por ejemplo: si un analizador quiere leer un dato de la cola de QoS, enviará a través del canal *socAnalizadores* el mensaje “READ_QoS” y el gestor de colas desencolará el primer elemento que haya en la cola de QoS y se lo enviará al analizador.