DISEÑO DEL PROYECTO 1 – MIDDLEWARE

CESAR ANDRÉS GARCÍA POSADA

DANIEL GARCÍA GARCÍA

JUAN CAMILO GUERRERO ALARCÓN

UNIVERSIDAD EAFIT

INGENIERÍA DE SISTEMAS

TÓPICOS ESPECIALES DE TELEMÁTICA

MEDELLÍN, ANTIOQUIA

2021

DISEÑO

Diagrama de arquitectura del sistema

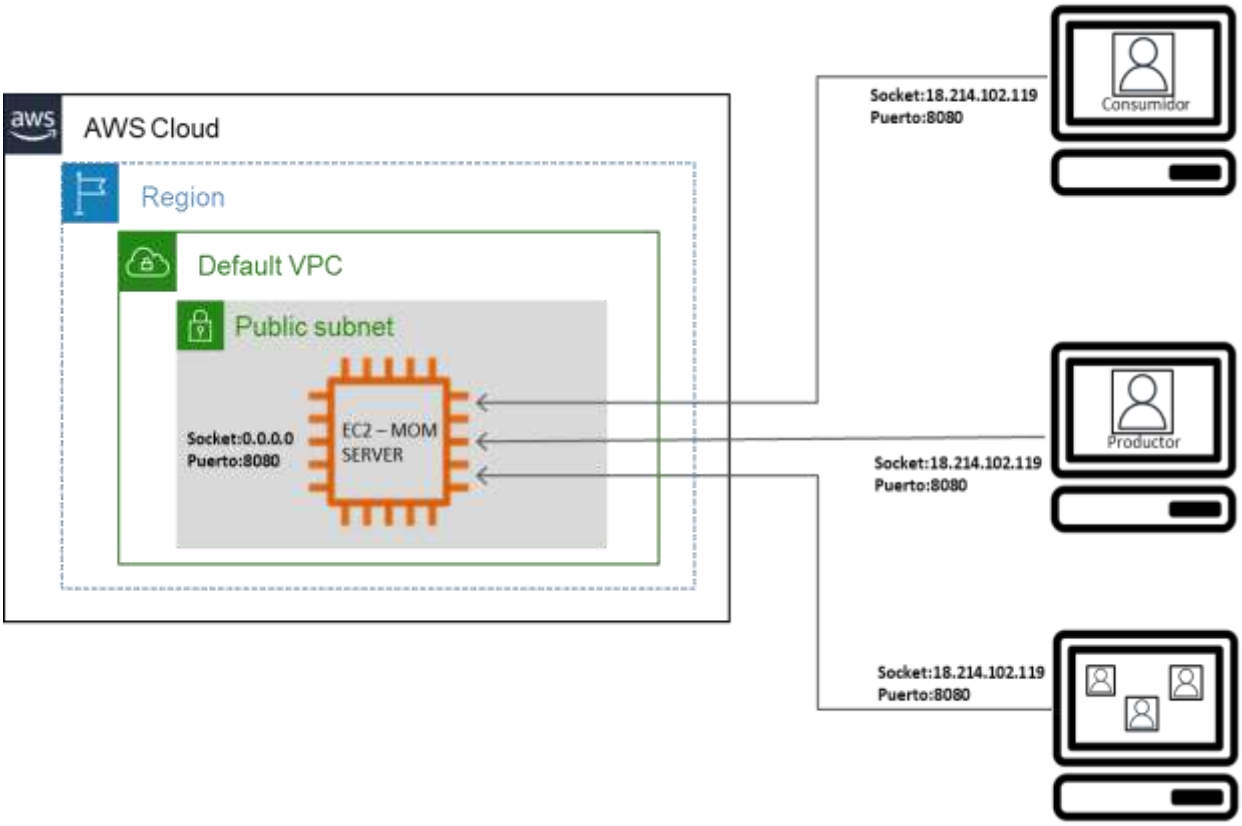

Figura 1. Diagrama de arquitectura del sistema

En la Figura 1, presentamos el diseño propuesto para la arquitectura del sistema distribuido, en este diagrama podemos observar, cómo se conectan los componentes del sistema para comunicarse mediante el MOM (Middleware orientado a mensajes), tenemos en un primer lugar el servidor MOM, que, para efectos de esta implementación, lo alojamos en una máquina EC2 de AWS haciendo uso de la computación en nube y todos los beneficios que esta nos provee. En segundo lugar, podemos observar que el sistema cumple con el atributo de escalabilidad, permitiendo la conexión de múltiples clientes desde sus máquinas locales, el usuario sólo necesitará ejecutar la aplicación cliente que desea utilizar (Productor o consumidor), dicha aplicación se conectará directamente con el server que gestiona el MOM haciendo uso de sockets y del puerto 8080.

Arquitectura de las colas

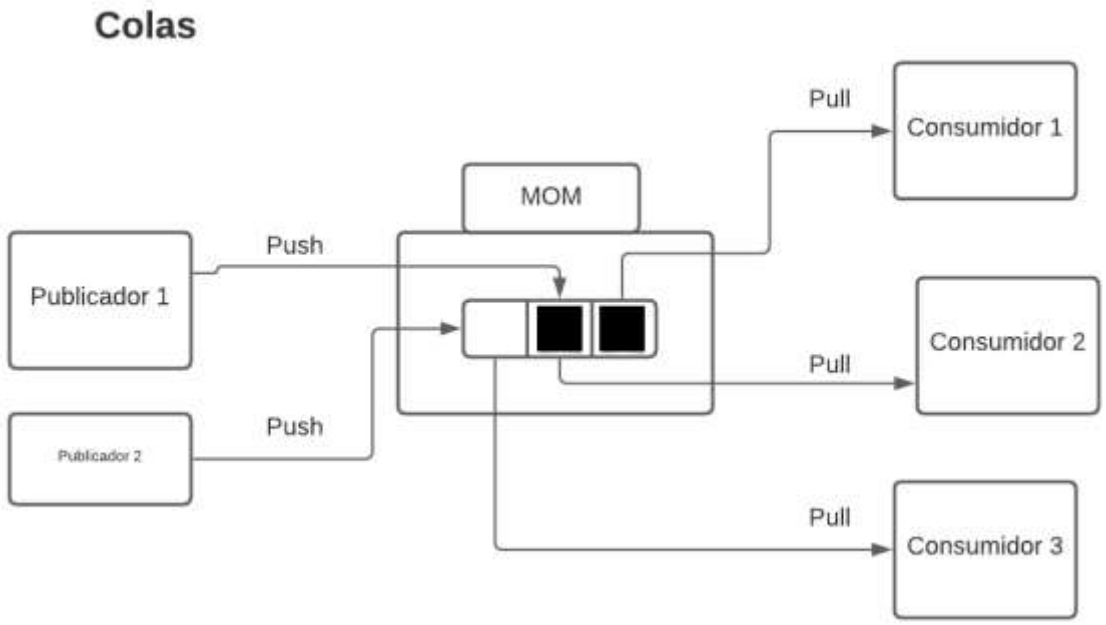

Figura 2. Diagrama de arquitectura para las colas

En la *Figura 2* podemos observar, la propuesta de arquitectura para el funcionamiento de las colas en el MOM, tenemos entonces que uno o más usuarios publicadores pueden agregar un mensaje a una cola donde se conectaron, los mensajes se van agregando, y los usuarios consumidores que se inscribieron a la misma cola, van entonces a recibir un mensaje específico(tarea). A medida que un usuario recibe una tarea, esta tarea o mensaje sale de la cola y el siguiente mensaje le queda al usuario siguiente que esté conectado a la cola. Los mensajes siempre se repartirán entre los usuarios consumidores conectados.

Arquitectura de los canales

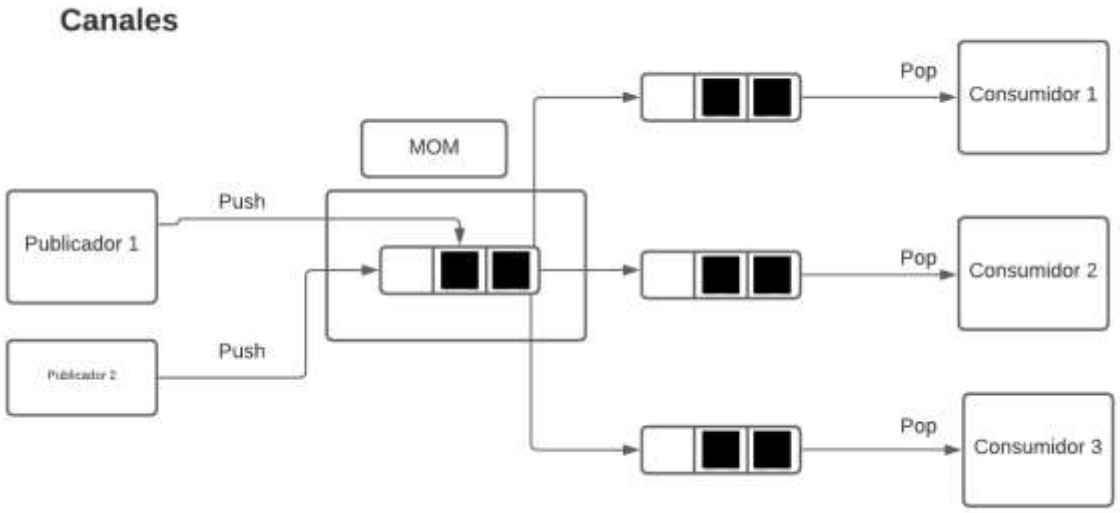

Figura 3. Diagrama de arquitectura para los canales

Para la *Figura 3* tenemos la propuesta de arquitectura para el funcionamiento de los canales en el MOM, podemos observar que aunque tenemos varios publicadores y varios consumidores conectados a un canal como en la arquitectura de las colas, hay un cambio sustancial, y es específicamente en la recepción de mensajes, los mensajes enviados se almacenan en una cola (como estructura de datos para la implementación), pero esta cola queda completamente disponible para cada uno de los usuarios consumidores conectados, siendo varios hilos de usuarios que acceden a la misma cola, y por tanto cada uno de ellos, puede acceder a todos los mensajes disponibles en el canal al cuál se conectaron, ya no acceden a un mensaje específico, sino que todos leen los mismos mensajes del canal.

Diagrama de clases

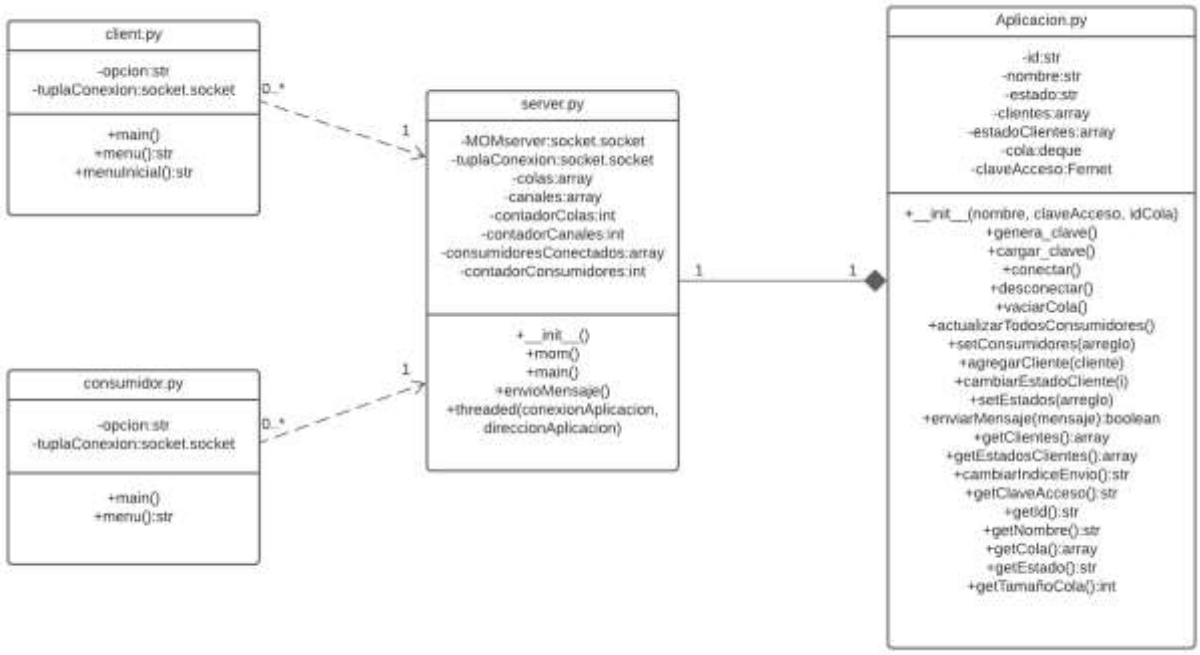

Figura 3. Diagrama de clases del Middleware

En el anterior diagrama de clases podemos observar los atributos y métodos más relevantes de cada clase que existe en el middleware, las funciones específicas que realiza cada clase las podemos entender de la siguiente manera:

- **client.py:** Gestiona las operaciones que puede hacer un usuario publicador mostrándole un menú de acciones, se comunica con el server como cliente para enviar peticiones y recibir respuestas, mediante sockets.

- **consumidor.py:** Gestiona las operaciones que puede hacer un usuario consumidor, mostrándole un menú de acciones, se comunica también como cliente con el server para enviar peticiones y recibir respuestas, mediante el uso sockets.

- **server2.py:** Servidor encargado de tomar decisiones de acuerdo a las opciones ingresadas por los usuarios, gestiona todo el MOM utilizando las funciones de Aplicación.py para posteriormente devolver la respuesta a las peticiones hechas por los clientes.

- **Aplicación.py:** Se encarga de definir los métodos para las acciones que se toman en el MOM cumpliendo los requerimientos del proyecto, tiene entonces la función de gestionar toda la información de las colas (arreglos bajo el modelo de cola o canal), los estados de estas anteriores, estados de los usuarios, los mensajes y encriptar las claves de acceso a los canales o colas.

En cuanto a las relaciones del diagrama, podemos ver que cada cliente tiene una relación de dependencia con el servidor, ya que le solicita mensajes a este y establece una comunicación, los resultados dependen del funcionamiento del server, además, existe un solo servidor que para efectos de escalabilidad puede establecer comunicación con muchos clientes tanto consumidores como productores. Tenemos una última relación desde el server hasta la clase Aplicación.py, que pusimos como una relación de composición, ya que el server hace parte de la aplicación, la utiliza para ejecutar las acciones que solicita un cliente y si Aplicación.py falla, la clase server2.py también lo hará.