Tomás Díaz 202220658

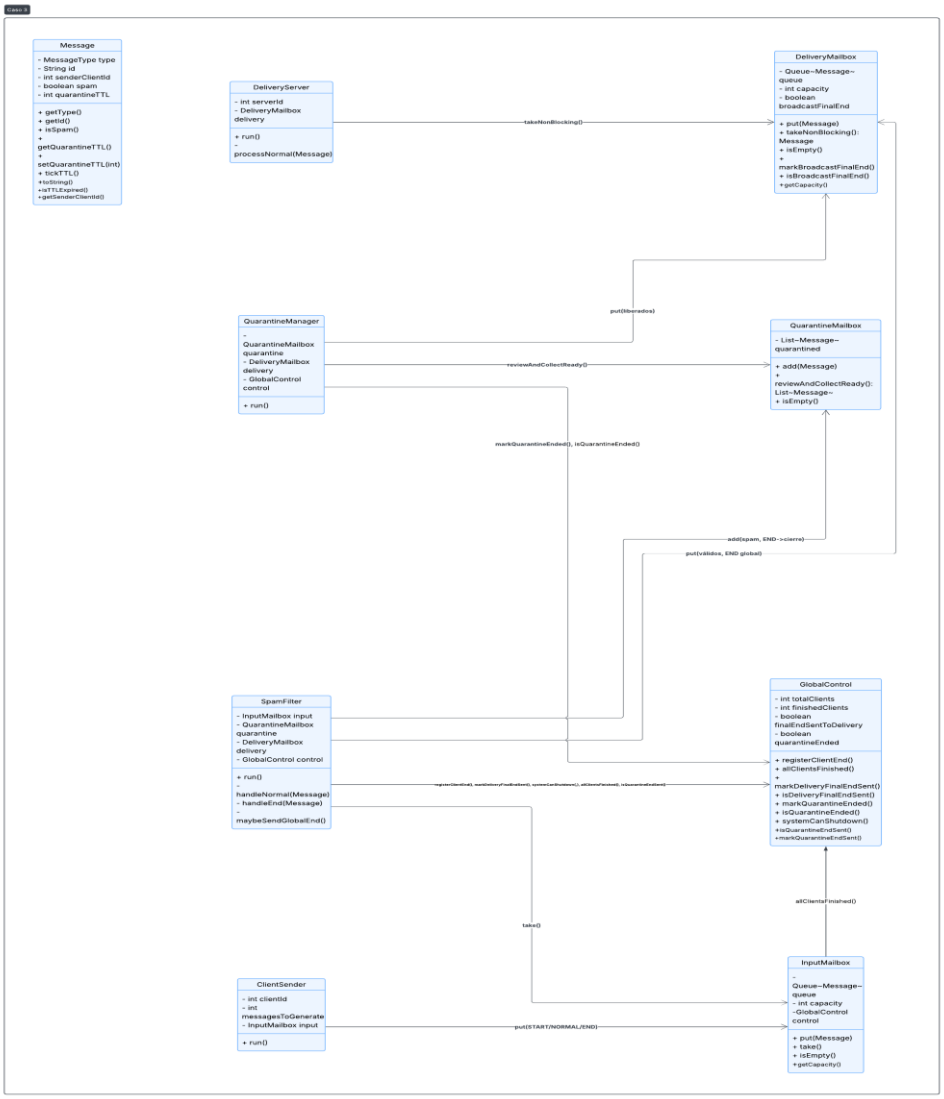
Juan Esteban Rojas 202124797

Informe explicando el diseño (diagrama de clase):

Se realizó un diagrama de clase, siguiendo la estructura de nuestro programa con sus atributos y servicios.

Si no se evidencia, se adjunta el pdf y link para revisar a profundidad.

https://lucid.app/lucidchart/983d43af-c252-4e80-b335-2c0491e10150/edit?viewport_loc=-788%2C1425%2C4881%2C2220%2C0_0&invitationId=inv_a71a989a-23c3-4474-86fe-7b8ff819adb5



Funcionamiento del programa:

El flujo de trabajo funciona: los clientes (ClientSender) crean los mensajes (Message) del correo y estos llegan al buzón Buzón de entrada (InputMailBox); que, si este está lleno, estos se bloquean, con una espera activa. Después, en este buzón salen directamente hacia los filtros de spam (SpamFilter (), y dependiendo de si tiene el atributo spam o no, se redirecciona con espera pasiva. Si tiene spam, se manda al buzón de cuarentena (QuarantineMailbox) y se pone a dormir un tiempo determinado antes de salir. Si no es considerado spam, se manda al buzón de entrega (DeliveryMailbox DeliveryMailbox).

Si este llega a Cuarentena, el manejador de cuarentena (Quarantinemanajer) genera un numero malicioso de 1-21 y si es múltiplo de 21 este se desecha, de lo contrario, se va restando 1s cada iteración para acabar el tiempo de ttl. Cuando este tiempo se acaba y no se descarta, el mensaje viaja al buzón de entrega en espera semiactiva.

Cuando nos encontramos en el buzón de entrega, hace una espera activa hacia los servidores de entrega (DeliveryServer). Y Aquí se acaba el proceso del mensaje.

Todo este flujo es manejado por un control global (GlobalControl), que marca cuando todos los clientes terminaron y se envió a END en la entrega y la cuarentena también acabo, marcando de esta manera que todos los servicios acabaron.

Sección de validación

Durante el proceso de prueba se insertaron diferentes mensajes de depuración (prints) para observar el comportamiento interno del sistema y detectar posibles errores o desórdenes en la ejecución. Conforme se fue depurando, se mantuvo una traza final clara que describe detalladamente las acciones de cada actor. Posteriormente, se realizaron ejecuciones variando los parámetros principales para verificar el cumplimiento de las restricciones de concurrencia y sincronización. Como ejemplo representativo, se empleó la configuración: **Clientes emisores:** 2, **Mensajes por cliente:** 2, **Filtros de spam:** 2, **Servidores de entrega:** 2, **Capacidad del buzón de entrada:** 1 y **Capacidad del buzón de entrega:** 1. Estos valores fueron elegidos por su simplicidad, ya que facilitan el seguimiento del flujo del sistema. La traza completa correspondiente a esta ejecución se incluye en el apéndice para su revisión.

Los **clientes emisores** operan conforme a lo establecido. Cada cliente inicia su ejecución enviando un mensaje de inicio (START), como se evidencia en la línea *"ClientSender-0 insertó [START | client=0 | id=cliente0-START | spam=false | TTL=0]"*. Posteriormente, generan sus mensajes normales con un indicador de spam aleatorio —por ejemplo, *"ClientSender-0 insertó [NORMAL*

| *client=0 | id=cliente0-msg1 | spam=true | TTL=0]"*— y finalizan enviando un mensaje de fin, como se observa en *"ClientSender-0 envió END y terminó."*. Además, cuando el buzón de entrada se encuentra lleno, los clientes esperan pasivamente, lo cual se refleja en las líneas *"ClientSender-1 espera (buzón lleno)"* y *"ClientSender-0 espera (buzón lleno)"*. Esto demuestra que la lógica de producción, sincronización y control de capacidad está correctamente implementada.

El **buzón de entrada** cumple adecuadamente su función de control de concurrencia, manteniendo la capacidad limitada y permitiendo que los clientes esperen hasta que los filtros liberen espacio. La interacción entre productores y consumidores se evidencia en la secuencia *"SpamFilter-0 tomó [START | client=0 | id=cliente0-START | spam=false | TTL=0]"* y *"SpamFilter-1 tomó [NORMAL | client=0 | id=cliente0-msg1 | spam=true | TTL=0]"*, que muestran cómo los filtros van extrayendo los mensajes para su procesamiento en cuanto están disponibles.

Los **filtros de spam** operan según las reglas definidas. Cuando detectan un mensaje de spam, lo envían a la cuarentena asignándole un tiempo de retención aleatorio, como se observa en *"SpamFilter-1 detectó SPAM [NORMAL | client=0 | id=cliente0-msg1 | spam=true | TTL=19719] (TTL=19719)"*. En cambio, los mensajes válidos son aprobados y enviados al buzón de entrega, tal como se indica en *"SpamFilter-1 aprobó [NORMAL | client=0 | id=cliente0-msg2 | spam=false | TTL=0] -> entrega"*. Además, los filtros reconocen los mensajes de inicio y fin de cada cliente, reflejado en *"SpamFilter-0 vio START de cliente 0"* y *"SpamFilter-1 tomó [END | client=1 | id=cliente1-END | spam=false | TTL=0]"*. También esperan en caso de que los buzones están vacios, como se ve en "S*pamFilter-1 espera (buzón vacío)*". Finalmente, uno de los filtros genera el mensaje de cierre global del sistema, evidenciado en la línea *"SpamFilter-0 envió GLOBAL_END al buzón de entrega"*, antes de finalizar su ejecución.

El **manejador de cuarentena** actúa revisando periódicamente los mensajes almacenados, reduciendo su tiempo de vida y trasladando los que expiran al buzón de entrega. Este comportamiento se aprecia en las líneas *"QuarantineManager revisa cuarentena..."* y *"QuarantineManager recibió END de cuarentena"*, que confirman la ejecución cíclica del proceso y su finalización ordenada al recibir el mensaje de fin.

El **buzón de entrega** recibe los mensajes aprobados por los filtros y los distribuye a los servidores. Esto se observa, por ejemplo, en *"DeliveryServer-1 recibió START"* y *"DeliveryServer-1 entregó [NORMAL | client=0 | id=cliente0-msg2 | spam=false | TTL=0]"*. Asimismo, se gestiona correctamente el mensaje de fin global (GLOBAL_END) que marca la finalización del flujo de mensajes válidos.

Los **servidores de entrega** procesan los correos válidos que van recibiendo y terminan su ejecución al recibir un mensaje de fin. Este comportamiento se confirma en las líneas *"DeliveryServer-1 recibió END -> termina"* y *"DeliveryServer-1 terminó (servidor de entrega)"*, que evidencian la correcta secuencia de procesamiento y finalización de los hilos consumidores.

Finalmente, la simulación muestra una **terminación ordenada y limpia del sistema**. Todos los hilos —clientes, filtros, manejador de cuarentena y servidores— concluyen su ejecución, y los buzones quedan vacíos. La línea *"Simulación finalizada correctamente. Todos los hilos terminaron y los buzones deberían estar vacíos."* confirma el cierre sincronizado del sistema, garantizando que no quedan mensajes pendientes y que cada componente cumple su ciclo de vida conforme a lo esperado.

Para cada pareja de objetos que interactúan, cómo se realiza la sincronización

Cliente (ClientSender)< -> Buzon entrada (InputMailBox):

Este funciona como un productor y un buffer, pues el cliente se encarga de colocar el mensaje (put) en la sección crítica del buzón mientras que el buzón no esté lleno. Si este llega a estar lleno, se manda a dormir con una espera pasiva con un wait() y este se levanta hasta que salga el mensaje del buzón(take()) y ya no esté lleno con un notifyAll().

Buzon de filtro(SpamFilter) <-> Buzon de entrada (InputMailBox):

El filtro de spam utiliza el take(); en otras palabras, va tomando los mensajes del

buzón

Buzon de filtro(SpamFilter) <-> Buzon de Cuarentena (QuarantineMailBox):

Si el buzon de filtro identifica que el mensaje es considerado un spam, agrega con el add() de la seccion critica pero no hay bloqueos por la espera ilimitada. Este hace una espera semiactiva cuando se va restando 1 en el clico y repitiendo las iteraciones (QuarantineManager.reviewAndCollectReady()). Devolviendo una lista de mensajes listos para entregar.

Manejador de cuarentena (QuarantineManager)<-> Buzon entrega (DeliveryMailbox):

Este funciona con una espera semiactiva, que se manda a dormir por un wait() de 10 milisegundos si el buzón llega a estar lleno, y si este llega a estar dormido cuando el buzón se desocupa, este se levanta con un notifyall()

Filtro de spam (SpamFilters) <-> Buzón de entrega (DeliveryMailBox):

Este funciona exactamente igual que el manejador de cuarentena con buzón de entrega, pues los dos invocan la misma función, que es la que hace la espera semiactiva.

Servidor de entrega (DeliveryServer) <-> Buzón de entrega (DeliveryMailbox):

Este realiza una espera activa que consiste en revisar hasta que haya un mensaje, realizándose con un while iterativo de: while (msg == null) {}

Funcionamiento global del sistema.

El sistema opera como una tubería concurrente de tres etapas. Los clientes emisores publican en el buzón de entrada (capacidad limitada) un mensaje START, una secuencia de correos NORMAL con bandera de spam aleatoria y, al final, un END; si el buzón está lleno, esperan de forma pasiva. Los filtros de spam consumen pasivamente de ese buzón: los correos marcados como spam se envían a cuarentena (ilimitada) con un TTL aleatorio y los válidos al buzón de entrega (capacidad limitada, inserción semiactiva). Los END de clientes se registran y también se remiten a cuarentena. El manejador de cuarentena recorre cada segundo los mensajes, decrementa TTL, descarta maliciosos y libera los restantes al buzón de entrega. Los servidores de entrega realizan espera activa, procesan NORMAL y finalizan al recibir END. Cuando todos los clientes terminaron y entrada y cuarentena están vacías, un filtro emite un GLOBAL_END al buzón de entrega para cerrar ordenadamente a todos los servidores. El sistema concluye cuando los filtros detectan la condición global de fin y todos los buzones quedan vacíos.

Apéndices

Print del sistema al realizar la prueba de explicación:

=== Configuración cargada ===

Clientes emisores: 2

Mensajes por cliente: 2

Filtros de spam: 2

Servidores de entrega: 2

Capacidad buzón entrada: 1

Capacidad buzón entrega: 1

==========

ClientSender-0 inició envío.

ClientSender-1 inició envío.

ClientSender-0 insertó [START | client=0 | id=cliente0-START | spam=false | TTL=0]

ClientSender-0 envió START.

ClientSender-1 espera (buzón lleno).

SpamFilter-0 tomó [START | client=0 | id=cliente0-START | spam=false | TTL=0]

ClientSender-0 insertó [NORMAL | client=0 | id=cliente0-msg1 | spam=true | TTL=0]

ClientSender-0 envió [NORMAL | client=0 | id=cliente0-msg1 | spam=true | TTL=0]

ClientSender-0 espera (buzón lleno).

SpamFilter-1 tomó [NORMAL | client=0 | id=cliente0-msg1 | spam=true | TTL=0]

ClientSender-1 insertó [START | client=1 | id=cliente1-START | spam=false | TTL=0]

ClientSender-1 envió START.

ClientSender-1 espera (buzón lleno).

ClientSender-0 espera (buzón lleno).

SpamFilter-1 detectó SPAM [NORMAL | client=0 | id=cliente0-msg1 | spam=true | TTL=19719] (TTL=19719)

SpamFilter-0 vio START de cliente 0

SpamFilter-1 tomó [START | client=1 | id=cliente1-START | spam=false | TTL=0]

SpamFilter-1 vio START de cliente 1

SpamFilter-1 espera (buzón vacío).

ClientSender-1 insertó [NORMAL | client=1 | id=cliente1-msg1 | spam=true | TTL=0]

ClientSender-0 espera (buzón lleno).

ClientSender-1 envió [NORMAL | client=1 | id=cliente1-msg1 | spam=true | TTL=0]

SpamFilter-1 tomó [NORMAL | client=1 | id=cliente1-msg1 | spam=true | TTL=0]

SpamFilter-1 detectó SPAM [NORMAL | client=1 | id=cliente1-msg1 | spam=true | TTL=19387] (TTL=19387)

ClientSender-0 insertó [NORMAL | client=0 | id=cliente0-msg2 | spam=false | TTL=0]

ClientSender-0 envió [NORMAL | client=0 | id=cliente0-msg2 | spam=false | TTL=0]

DeliveryServer-1 recibió START

SpamFilter-1 tomó [NORMAL | client=0 | id=cliente0-msg2 | spam=false | TTL=0]

SpamFilter-1 aprobó [NORMAL | client=0 | id=cliente0-msg2 | spam=false | TTL=0] -> entrega

ClientSender-0 insertó [END | client=0 | id=cliente0-END | spam=false | TTL=0]

ClientSender-0 envió END y terminó.

ClientSender-1 espera (buzón lleno).

DeliveryServer-0 recibió START

SpamFilter-0 tomó [END | client=0 | id=cliente0-END | spam=false | TTL=0]

SpamFilter-0 espera (buzón vacío).

ClientSender-1 insertó [NORMAL | client=1 | id=cliente1-msg2 | spam=false | TTL=0]

ClientSender-1 envió [NORMAL | client=1 | id=cliente1-msg2 | spam=false | TTL=0]

SpamFilter-0 tomó [NORMAL | client=1 | id=cliente1-msg2 | spam=false | TTL=0]

SpamFilter-0 aprobó [NORMAL | client=1 | id=cliente1-msg2 | spam=false | TTL=0] -> entrega

ClientSender-1 insertó [END | client=1 | id=cliente1-END | spam=false | TTL=0]

ClientSender-1 envió END y terminó.

SpamFilter-1 tomó [END | client=1 | id=cliente1-END | spam=false | TTL=0]

SpamFilter-1 envió único END a cuarentena.

DeliveryServer-1 entregó [NORMAL | client=0 | id=cliente0-msg2 | spam=false | TTL=0]

DeliveryServer-1 recibió END -> termina

DeliveryServer-1 terminó (servidor de entrega).

SpamFilter-0 envió GLOBAL_END al buzón de entrega

DeliveryServer-0 entregó [NORMAL | client=1 | id=cliente1-msg2 | spam=false | TTL=0]

DeliveryServer-0 recibió END -> termina

DeliveryServer-0 terminó (servidor de entrega).

QuarantineManager revisa cuarentena...

QuarantineManager recibió END de cuarentena.

SpamFilter-1 terminó (filtro de spam).

SpamFilter-0 terminó (filtro de spam).

QuarantineManager terminó (manejador de cuarentena).

Simulación finalizada correctamente.

Todos los hilos terminaron y los buzones deberían estar vacíos.