Trabajo Práctico Especial 1
Mostradores de Check-in

10 de Abril de 2024

Objetivo

Implementar en grupos un **sistema remoto *thread-safe*** para la **asignación de mostradores de check-in de un aeropuerto** permitiendo notificar a las aerolíneas y ofreciendo reportes de los check-ins realizados hasta el momento.

Introducción

Horas antes de que un vuelo de pasajeros despegue de un aeropuerto, en los mostradores de check-in el personal de la aerolínea se encarga de recibir a los pasajeros que cuentan con un ticket para ese vuelo, para verificar sus datos, recibir su equipaje y realizar el check-in para así permitirles embarcar e iniciar su viaje. Este trabajo práctico cubre los aspectos fundamentales del proceso de asignación de mostradores de check-in desde el punto de vista del ***pasajero***, las ***aerolíneas*** y la ***administración del aeropuerto*** a través de la implementación de cinco servicios remotos.

Un ***pasajero*** se acerca al aeropuerto con su código de reserva. En caso de que ya existan mostradores que estén esperando pasajeros de ese vuelo se le indica el rango de mostradores correspondiente y el pasajero se acerca para ingresar a una cola única para todo el rango. Las aerolíneas cuentan con varios mostradores juntos, llamado rango, para así atender a más

pasajeros en simultáneo. Una vez ingresado a la cola, espera su turno, y cuando este llega realiza el check-in y puede embarcar.

Las ***aerolíneas*** deben solicitar a la administración del aeropuerto el rango de mostradores para iniciar el proceso de check-in. Como la cantidad de mostradores del aeropuerto es limitada es común que una aerolínea quiera solicitar una cantidad de mostradores y estos no estén disponibles porque están siendo utilizados para otros vuelos. Entonces se registra el pedido de la aerolínea y cuando los mostradores se desocupen o se agreguen nuevos se podrá intentar resolver la solicitud pendiente. Una vez asignados, cuando la aerolínea termina el proceso se encarga de liberar los mostradores que le fueron asignados para que éstos sean utilizados por otros vuelos/aerolíneas.

La ***administración del aeropuerto*** divide al mismo en sectores y se encarga de ir creando mostradores ubicándolos en un sector determinado. Además la administración recibe el manifiesto con todos los pasajeros que se esperan que embarquen en el aeropuerto en función de los tickets que hayan vendido las aerolíneas. Por último debe poder consultar todo lo ocurrido en cada uno de los mostradores.

Servicios Remotos y Clientes

El sistema requiere el desarrollo de **los siguientes servicios remotos con sus respectivos clientes**.

Es importante destacar que, aunque en este enunciado se mencionan servicios y clientes de manera conjunta por motivos de claridad y brevedad, es importante que **la implementación respete la correcta separación entre servicio y cliente**. Para ello analizar si la lógica a implementar corresponde que esté del lado del servicio, del cliente o de ambas. Considerar que un mismo servicio puede ser invocado por varios clientes (por ejemplo un cliente de línea de comandos y un cliente de una aplicación mobile). En este enunciado nos limitaremos a implementar un único cliente de línea de comandos para cada uno de los servicios presentados.

1. Servicio de Administración del Aeropuerto

- Funcionalidad: Agregar sectores, agregar mostradores y actualizar el listado de aerolíneas y pasajeros esperados en el aeropuerto.
- Usuario: Empresa Administradora del Aeropuerto
- Cliente: La información de cuál es la acción a realizar se recibe a través de argumentos de línea de comando al llamar al script del cliente de administración del aeropuerto **`adminClient.sh`** y el resultado se debe imprimir en pantalla.

```
$> sh adminClient.sh -DserverAddress=xx.xx.xx.xx:yyyy -Daction=actionName
[ -Dsector=sectorName | -Dcounters=counterCount | -DinPath=manifestPath ]
```

donde

- `xx.xx.xx.xx:yyyy` es la dirección IP y el puerto donde está publicado el servicio de administración del aeropuerto
- `actionName` es el nombre de la acción a realizar (que se detallan en las siguientes secciones)

72.42 Programación de Objetos Distribuidos

1.1 Agregar un sector

- Funcionalidad: Agrega un sector a partir de un nombre `sectorName`. El sector inicia sin mostradores.
- Falla si:
  - Ya existe un sector con ese nombre
- Ejemplo de cliente de invocación: Agregar el sector “C” al aeropuerto

```
$> sh adminClient.sh -DserverAddress=10.6.0.1:50051 -Daction=addSector
-Dsector=C
Sector C added successfully
```

1.2 Agregar un rango de mostradores

- Funcionalidad: Agrega un rango de mostradores contiguos en un sector a partir del nombre del sector `sectorName` y la cantidad de mostradores `counterCount` a agregar. Los mostradores inician sin asignaciones. La numeración de los mostradores es incremental y compartida entre todos los sectores del aeropuerto donde el primero tiene el número 1. Por lo tanto, para un sector, la numeración de los mostradores puede contener *gaps*.
- Falla si:
  - No existe un sector con el nombre indicado
  - La cantidad de mostradores indicada no es positiva
- Ejemplo de cliente de invocación: Agregar “3” mostradores en el sector “C” del aeropuerto después de haber agregado un mostrador en otro sector.

```
$> sh adminClient.sh -DserverAddress=10.6.0.1:50051 -Daction=addCounters
-Dsector=C -Dcounters=3
3 new counters (2-4) in Sector C added successfully
```

1.3 Agregar un pasajero esperado

- Funcionalidad: Agregar un pasajero que debe embarcar en el aeropuerto a partir del código de la reserva, el código del vuelo y el nombre de la aerolínea.
- Falla si:
  - Ya se agregó un pasajero con ese código de reserva
  - Ya se agregó un vuelo con ese código pero con otra aerolínea
- Ejemplo de cliente invocación: Agregar un **lote** de pasajeros a partir de un archivo CSV `manifestPath` con la siguiente estructura:
  - `booking`: código de la reserva (alfanumérico de 6 caracteres)
  - `flight`: código del vuelo
  - `airline`: nombre de la aerolínea

  donde cada línea representa un pasajero esperado en el aeropuerto

  Por ejemplo para un archivo `manifest.csv` ubicado en el directorio superior al script y con siguiente contenido:

```
booking;flight;airline
ABC123;AC987;AirCanada
XYZ234;AA123;AmericanAirlines
...
```

para la segunda línea se agrega al pasajero con código de reserva “ABC123” para el vuelo “AC987” de “AirCanada”

```
$> sh adminClient.sh -DserverAddress=10.6.0.1:50051 -Daction=manifest
-DinPath=../manifest.csv
    Booking ABC123 for AirCanada AC987 added successfully
    ...
```

2. Servicio de Reserva de Mostradores

- Funcionalidad: Consultar los sectores, mostradores y solicitar y liberar un rango de mostradores para luego realizar check-ins de los vuelos de una aerolínea
- Usuario: Aerolínea que tiene uno o más vuelos con partidas en el aeropuerto
- Cliente: La información de cuál es la acción a realizar se recibe a través de argumentos de línea de comando al llamar al script del cliente de reserva de mostradores **counterClient.sh** y el resultado se debe imprimir en pantalla.

```
$> sh counterClient.sh -DserverAddress=xx.xx.xx.xx:yyyy
-Daction=actionName [ -Dsector=sectorName | -DcounterFrom=fromVal |
-DcounterTo=toVal | -Dflights=flights | -Dairline=airlineName |
-DcounterCount=countVal ]
```

donde

- xx.xx.xx.xx:yyyy es la dirección IP y el puerto donde está publicado el servicio de reserva de mostradores
- actionName es el nombre de la acción a realizar (que se detallan en las siguientes secciones)

2.1 Consultar los sectores

- Funcionalidad: Consultar los sectores y los rangos de mostradores ubicados en cada uno de ellos, en orden alfabético por sector
- Falla si:
  - No existen sectores en el aeropuerto
- Ejemplo de cliente de invocación: Consultar los sectores, donde el sector A tiene 1 solo mostrador, el sector C tiene 5 mostradores, el sector D tiene 2 mostradores y el sector Z no tiene mostradores

```
$> sh counterClient.sh -DserverAddress=10.6.0.1:50051
-Daction=listSectors
    Sectors   Counters
    ###################
    A         (1-1)
    C         (2-4)(7-8)
    D         (5-6)
    Z         -
```

2.2 Consultar un rango de mostradores

- Funcionalidad: Consultar un rango de mostradores [`fromVal`, `toVal`] de un sector `sectorName`, en orden ascendente por mostrador, indicando si el mostrador está libre o si está asignado a una aerolínea (en ese caso indicar los códigos de los vuelos de la asignación y la cantidad de personas esperando a ser atendidas en la cola del rango de mostradores)
- Falla si:
  - No existe un sector con ese nombre
  - `fromVal` y `toVal` no forman un rango de uno o más mostradores
- Ejemplo de cliente de invocación: Consultar los mostradores “2” a “5” inclusive del sector “C” indicando que los mostradores 2 y 3 están siendo usados por American Airlines para atender a pasajeros de los vuelos AA123, AA124 y AA125 con 6 personas en la cola del rango esperando a ser atendidas, que el mostrador 4 está libre y que el mostrador 5 no existe en el sector.

```
$> sh counterClient.sh -DserverAddress=10.6.0.1:50051
-Daction=listCounters -Dsector=C -DcounterFrom=2 -DcounterTo=5
   Counters  Airline          Flights             People
   ############################################################
   (2-3)     AmericanAirlines AA123|AA124|AA125   6
   (4-4)     -                -                   -
```

En el caso de que el rango de mostradores sea vacío o si se consulta un sector sin mostradores

```
$> sh counterClient.sh -DserverAddress=10.6.0.1:50051
-Daction=listCounters -Dsector=C -DcounterFrom=10 -DcounterTo=20
   Counters  Airline          Flights             People
   ############################################################
```

2.3 Asignar un rango de mostradores

- Funcionalidad: Asignar un rango de `countVal` mostradores contiguos ubicados en el sector `sectorName` para que los pasajeros puedan realizar el check-in de los vuelos `flights` de la aerolínea `airlineName`. Para el **algoritmo determinístico** de asignación se deben respetar los siguientes criterios:
  - Prioridad: Considerar primero los mostradores con identificadores más bajos
  - Contigüidad: Si por ejemplo en un sector se tienen 5 mostradores del 1 al 5 donde los mostradores 2 y 3 están en uso, si se quiere solicitar un rango de 3 mostradores no es posible asignarlo porque si bien están disponibles los mostradores 1, 4 y 5 estos no están contiguos.
  - **Asignación Pendiente:** En caso de no poder realizar la asignación deberá almacenar la asginación en estado pendiente.

    Con la invocación a alguno de los siguientes métodos se deberán intentar resolver todas las asignaciones pendientes correspondientes al sector indicado, respetando el orden en que las aerolíneas lo solicitaron:
    i. **1.2 Agregar un Rango de Mostradores**
    ii. **2.4 Liberar un Rango de Mostradores**

Las asignaciones que no puedan realizarse se mantendrán en estado pendiente preservando el orden en que fueron solicitadas. En ese caso se debe indicar además cuantas solicitudes pendientes existen actualmente.

- Falla si:
    - No existe un sector con ese nombre
    - No se agregaron pasajeros esperados con el código de vuelo, para al menos uno de los vuelos solicitados
    - Se agregaron pasajeros esperados con el código de vuelo pero con otra aerolínea, para al menos uno de los vuelos solicitados
    - Ya existe al menos un mostrador asignado para al menos uno de los vuelos solicitados (no se permiten agrandar rangos de mostradores asignados)
    - Ya existe una solicitud pendiente de un rango de mostradores para al menos uno de los vuelos solicitados (no se permiten reiterar asignaciones pendientes)
    - Ya se asignó y luego se liberó un rango de mostradores para al menos uno de los vuelos solicitados (no se puede iniciar el check-in de un vuelo dos o más veces)
- Ejemplo de cliente de invocación: Solicitar un rango de “2” mostradores para los check-ins de los vuelos “AA123”, “AA124” y “AA125” de “AmericanAirlines” en el sector “C” de forma que el rango asignado incluye los mostradores 3 y 4.

```
$> sh counterClient.sh -DserverAddress=10.6.0.1:50051
-Daction=assignCounters -Dsector=C -Dflights=AA123|AA124|AA125
-Dairline=AmericanAirlines -DcounterCount=2
        2 counters (3-4) in Sector C are now checking in passengers from
AmericanAirlines AA123|AA124|AA125 flights
```

En caso de que la asignación no se haya podido realizar se indica que la misma se almacena en estado pendiente y que hay otras 5 solicitudes pendientes antes de ella.

```
$> sh counterClient.sh -DserverAddress=10.6.0.1:50051
-Daction=assignCounters -Dsector=C -Dflights=AA123|AA124|AA125
-Dairline=AmericanAirlines -DcounterCount=2
        2 counters in Sector C is pending with 5 other pendings ahead
```

2.4 Liberar un rango de mostradores

- Funcionalidad: Liberar un rango de mostradores que inicia en `fromVal` del sector `sectorName` actualmente asignados para vuelos de la aerolínea `airlineName` de forma que los pasajeros ya no podrán realizar el check-in
- Falla si:
    - No existe un sector con ese nombre
    - El rango de mostradores no existe en ese sector (los mostradores no están asignados)
    - El rango de mostradores existe pero no corresponde a esa aerolínea (una aerolínea sólo puede liberar sus rangos)
    - Existen pasajeros esperando a ser atendidos en la cola del rango
- Ejemplo de cliente de invocación: Liberar el rango de mostradores que inicia en “3” del sector “C” que actualmente está en uso por “AmericanAirlines”

```
$> sh counterClient.sh -DserverAddress=10.6.0.1:50051
-Daction=freeCounters -Dsector=C -DcounterFrom=3 -Dairline=AmericanAirlines
```

```
        Ended check-in for flights AA123|AA124|AA125 on 2 counters (3-4) in
Sector C
```

2.5 Realizar un check-in para cada mostrador

- Funcionalidad: Realizar el check-in de un pasajero que está en la cola de un rango de mostradores para cada uno de los mostradores del rango que inicia en `fromVal` del sector `sectorName` actualmente asignados para vuelos de la aerolínea `airlineName`. El **algoritmo determinístico** de check-in consiste en que cada mostrador del rango atienda a un pasajero que está en la cola. Para el algoritmo se deben respetar los siguientes criterios:
    - Prioridad: Considerar primero los mostradores con identificadores más bajos
    - Todos los mostradores de un rango realizan el check-in de 0 o 1 pasajeros con una sola invocación al método respetando el orden de ingreso a la cola: Si por ejemplo se cuenta con un rango de 3 mostradores del 4 al 6 inclusive y existen 5 pasajeros esperando en la cola y se invoca al método entonces se realizarán 3 check-ins. En el mostrador 4 se realiza el check-in del pasajero que estaba primero en la cola. En el mostrador 5 se realiza el check-in del segudo pasajero que ingresó en la cola y por último en el mostrador 6 se realiza el check-in del tercer pasajero en la cola. En una siguiente invocación al método los mostradores 4 y 5 realizarán el check-in de los pasajeros que quedaron en la cola y el mostrador 6 no realiza ningún check-in.
    - Los pasajeros ingresan a la cola a partir de la invocación al método **3.2. Ingresar a la cola de un rango de mostradores** del **servicio de Check-in de Pasajeros**.
- Falla si:
    - No existe un sector con ese nombre
    - El rango de mostradores no existe en ese sector (los mostradores no están asignados)
    - El rango de mostradores existe pero no corresponde a esa aerolínea
- Ejemplo de cliente de invocación: Realizar check-ins en el rango de mostradores que inicia en “3” del sector “C” para la aerolínea “AmericanAirlines” de forma que como en la cola había sólo un pasajero con la reserva XYZ345 se realiza el check-in con éxito en el mostrador “3” y en el mostrador “4” no se realiza ningún check-in.

```
    $> sh counterClient.sh -DserverAddress=10.6.0.1:50051
-Daction=checkinCounters -Dsector=C -DcounterFrom=3 -Dairline=AmericanAirlines
        Check-in successful of XYZ345 for flight AA123 at counter 3
        Counter 4 is idle
```

2.6 Consultar las asignaciones pendientes

- Funcionalidad: Consultar las asignaciones pendientes de rangos de mostradores de un sector `sectorName`, respetando el orden en que fueron realizadas, indicando la cantidad de mostradores del rango, la aerolínea que lo solicitó y los códigos de los vuelos de cada asignación
- Falla si:
    - No existe un sector con ese nombre
- Ejemplo de cliente de invocación: Consultar las asignaciones pendientes del sector “C” indicando que la asignación pendiente más antigua es la de un rango de 2 mostradores de AirCanada para el vuelo AC003.

72.42 Programación de Objetos Distribuidos

```
      $> sh counterClient.sh -DserverAddress=10.6.0.1:50051
-Daction=listPendingAssignments -Dsector=C
         Counters  Airline          Flights
         ############################################################
         2         AirCanada        AC003
         5         AmericanAirlines AA987|AA988
         2         AirCanada        AC001
```

3. Servicio de Check-in de Pasajeros

- Funcionalidad: Conocer el rango de mostradores donde puede realizar el check-in y realizar el check-in
- Usuario: Pasajero que debe embarcar en el aeropuerto
- Cliente: La información de cuál es la acción a realizar se recibe a través de argumentos de línea de comando al llamar al script del cliente de check-in de pasajeros **passengerClient.sh** y el resultado se debe imprimir en pantalla.

```
      $> sh passengerClient.sh -DserverAddress=xx.xx.xx.xx:yyyy
-Daction=actionName [ -Dbooking=booking | -Dsector=sectorName |
-Dcounter=counterNumber ]
```

donde

- xx.xx.xx.xx:yyyy es la dirección IP y el puerto donde está publicado el servicio de check-in de pasajeros
- actionName es el nombre de la acción a realizar (que se detallan en las siguientes secciones)

3.1 Obtener el rango de mostradores asignado para check-in

- Funcionalidad: Consultar el rango de mostradores asignado para realizar el check-in de un vuelo a partir del código de reserva booking, indicando la cantidad de pasajeros esperando en la cola de ese rango y el sector en donde se encuentra.
- Falla si:
  - No existe un pasajero esperado con ese código de reserva
- Ejemplo de cliente de invocación: Consultar el rango de mostradores asignado para realizar el check-in con el código de reserva “XYZ123” de forma que se indica que se puede acercar al rango de mostradores del sector “C” que inicia en “3”, que cuenta con 2 mostradores y donde hay 7 personas en la cola del rango.

```
      $> sh passengerClient.sh -DserverAddress=10.6.0.1:50051
-Daction=fetchCounter -Dbooking=XYZ345
         Flight AA123 from AmericanAirlines is now checking in at counters
(3-4) in Sector C with 7 people in line
```

En caso de que invoque con un código de reserva correspondiente a un vuelo que todavía no tiene ninguna asignación (no hay mostradores asignados que atiendan pasajeros de ese vuelo) mostrar el siguiente mensaje

```
$> sh passengerClient.sh -DserverAddress=10.6.0.1:50051
-Daction=fetchCounter -Dbooking=XYZ345
    Flight AA123 from AmericanAirlines has no counters assigned yet
```

3.2 Ingresar a la cola del rango de mostradores

- Funcionalidad: Ingresar a la cola de un rango de mostradores para realizar el check-in de un vuelo a partir del número de mostrador `counterNumber` correspondiente al inicio del rango, el nombre del sector `sectorName` y el código de reserva `booking`, indicando además la cantidad de pasajeros esperando en la cola de ese rango.
- Falla si:
    - No existe una reserva con ese código
    - No existe un sector con ese nombre
    - El número de mostrador no corresponde con el inicio de un rango de mostradores asignado a la aerolínea que esté aceptando pasajeros del vuelo de la reserva
    - El pasajero ya ingresó en la cola del rango
    - El pasajero ya realizó el check-in de la reserva
- Ejemplo de cliente de invocación: Ingresa a la cola del rango de mostradores que inicia en "3" en el sector "C" para realizar el check-in con el código de reserva "ABC123" y hay 6 personas delante esperando a ser atendidas. El check-in se confirmará luego de una o varias invocaciones al método **2.5 Realizar un check-in para cada mostrador** del **Servicio de Reserva de Mostradores.**

```
$> sh passengerClient.sh -DserverAddress=10.6.0.1:50051
-Daction=passengerCheckin -Dbooking=ABC123 -Dsector=C -Dcounter=3
    Booking ABC123 for flight AA123 from AmericanAirlines is now waiting
to check-in on counters (3-4) in Sector C with 6 people in line
```

3.3 Consultar el estado de check-in

- Funcionalidad: Consultar el estado de check-in de un pasajero a partir del código de reserva `booking`, indicando:
    - El número de mostrador y nombre del sector si ya realizó el check-in
    - El rango de mostradores, nombre del sector y la cantidad de personas que tiene delante si ingresó en la cola
    - El rango de mostradores y nombre del sector si todavía no ingresó a la cola
- Falla si:
    - No existe un pasajero esperado con ese código de reserva
    - No hay un rango de mostradores asignados que atiendan pasajeros del vuelo correspondiente al código de reserva indicado
- Ejemplo de cliente de invocación: Consultar el estado de check-in con el código de reserva "ABC123" indicando que se realizó el check-in con éxito en el mostrador "4" del Sector "C".

```
$> sh passengerClient.sh -DserverAddress=10.6.0.1:50051
-Daction=passengerStatus -Dbooking=ABC123
    Booking ABC123 for flight AA123 from AmericanAirlines checked in at
counter 4 in Sector C
```

Si está esperando en la cola para hacer el check-in

Booking ABC123 for flight AA123 from AmericanAirlines is now waiting to check-in on counters (3-4) in Sector C with 6 people in line

Y si todavía no ingresó en la cola

Booking ABC123 for flight AA123 from AmericanAirlines can check-in on counters (3-4) in Sector C

4. Servicio de Notificaciones de Aerolínea

- Funcionalidad: Recibir notificaciones respecto a las asignaciones de mostradores de una aerolínea y los check-ins realizados en ellos
- Usuario: Aerolínea que tiene uno o más vuelos con partidas en el aeropuerto
- Cliente: La información de cuál es la acción a realizar se recibe a través de argumentos de línea de comando al llamar al script del cliente de notificaciones de aerolínea **eventsClient.sh** y el resultado se debe imprimir en pantalla.

`$> sh` **`eventsClient.sh`** `-DserverAddress=xx.xx.xx.xx:yyyy -Daction=actionName -Dairline=airlineName`

donde

- `xx.xx.xx.xx:yyyy` es la dirección IP y el puerto donde está publicado el servicio de notificaciones de aerolínea
- `actionName` es el nombre de la acción a realizar (que se detallan en las siguientes secciones)

4.1 Registrar a una aerolínea para ser notificada

- Funcionalidad: Registrar a una aerolínea `airlineName` para que sea notificada de los eventos relacionados a sus asignaciones de mostradores y los check-ins realizados en los mismos
  Se considera un evento:
  - Se **registró correctamente** para recibir notificaciones

AmericanAirlines registered successfully for check-in events

  - Se le **asignó** a la aerolínea un rango de mostradores o una **asignación pendiente se concretó**, indicando los mostradores involucrados, el sector correspondiente y los códigos de vuelos aceptados

2 counters (3-4) in Sector C are now checking in passengers from AmericanAirlines AA123|AA124|AA125 flights

  - Un **pasajero ingresó a la cola** de un rango de mostradores asignado a la aerolínea, indicando los mostradores involucrados, el sector correspondiente, los códigos de vuelos aceptados y la cantidad de pasajeros esperando en la cola

Booking ABC123 for flight AA123 from AmericanAirlines is now waiting to check-in on counters (2-4) in Sector C with 6 people in line

- Un mostrador correspondiente a un rango de mostradores asignado a la aerolínea **realizó check-in** de un pasajero indicando el código de la reserva, el sector correspondiente y el código de vuelo

Check-in successful of XYZ345 for flight AA123 at counter 3 in Sector C

- Se **liberó** un rango de mostradores asignado a la aerolínea, indicando los mostradores involucrados, el sector correspondiente y los códigos de vuelos aceptados

Ended check-in for flights AA123|AA124|AA125 on counters (2-4) from Sector C

- Se intentó asignar a la aerolínea un rango de mostradores y la **asignación** se almacena en estado **pendiente**

7 counters in Sector C for flights AA888|AA999 is pending with 5 other pendings ahead

- Cambió la cantidad de **asignaciones pendientes delante** de una asignación pendiente de la aerolínea

7 counters in Sector C for flights AA888|AA999 is pending with 4 other pendings ahead

- Falla si:
  - No existe al menos un pasajero esperado con un vuelo correspondiente a la aerolínea
  - La aerolínea ya se registró para recibir notificaciones
- Ejemplo de cliente de invocación: Registrar a la aerolínea “AmericanAirlines” para recibir notificaciones, mostrando el evento inicial de registro, donde el cliente no termina y muestra las notificaciones a medida que van ocurriendo

```
$> sh eventsClient.sh -DserverAddress=10.6.0.1:50051 -Daction=register -Dairline=AmericanAirlines
    AmericanAirlines registered successfully for events
    ...
```

4.2 Anular el registro de una aerolínea

- Funcionalidad: Anular el registro de una aerolínea `airlineName` para no recibir más notificaciones. Además finaliza la ejecución del cliente de notificaciones de esa aerolínea.
- Falla si:
  - La aerolínea no se registró para recibir notificaciones
- Ejemplo de invocación: Anular el registro de la aerolínea “AmericanAirlines” para no recibir más notificaciones

```
$> sh eventsClient.sh -DserverAddress=10.6.0.1:50051 -Daction=unregister -Dairline=AmericanAirlines
```

```
AmericanAirlines unregistered successfully for events
```

4.3 Consultar el historial de notificaciones de una aerolínea (***Exclusivo para el grupo de 5 integrantes***)

- Funcionalidad: Listar todas los eventos hasta el momento correspondientes a una aerolínea `airlineName` (desde el inicio del servicio).
- Falla si:
  - No existe al menos un pasajero esperado con un vuelo correspondiente a la aerolínea
- Ejemplo de cliente de invocación: Consultar el historial de notificaciones de la aerolínea “AmericanAirlines”.

```
$> sh eventsClient.sh -DserverAddress=10.6.0.1:50051 -Daction=history -Dairline=AmericanAirlines
2 counters (3-4) in Sector C are now checking in passengers from AmericanAirlines AA123|AA124|AA125 flights
7 counters in Sector C for flights AA888|AA999 is pending with 5 other pendings ahead
Check-in successful of XYZ345 for flight AA123 at counter 3 in Sector C
...
Ended check-in for flights AA123|AA124|AA125 on counters (2-4) from Sector C
```

5. Servicio de Consulta de Mostradores

- Funcionalidad: Consultar el estado actual de los mostradores y el historial de pasajeros que realizaron check-in
- Usuario: Empresa Administradora del Aeropuerto
- Cliente: La información de cuál es la acción a realizar se recibe a través de argumentos de línea de comando al llamar al script del cliente de consulta de mostradores **`queryClient.sh`** y el resultado se debe imprimir en pantalla.

```
$> sh queryClient.sh -DserverAddress=xx.xx.xx.xx:yyyy -Daction=actionName -DoutPath=query.txt [ -Dsector=sectorName | -Dairline=airlineName | -Dcounter=counterVal ]
```

donde

- `xx.xx.xx.xx:yyyy` es la dirección IP y el puerto donde está publicado el servicio de consulta de mostradores
- `actionName` es el nombre de la ***query*** a realizar (que se detallan en las siguientes secciones)
- `query.txt` es el *path* del archivo de salida con los resultados de la consulta elegida

5.1 Consultar el estado de los mostradores, filtrando por sector

- Funcionalidad: Consultar el estado actual de los mostradores, pudiendo listar únicamente los mostradores del sector `sectorName` con un parámetro opcional, indicando para cada

rango de mostradores el sector, los mostradores involucrados, el nombre de la aerolínea, los vuelos que acepta y la cantidad de personas esperando a ser atendidas en ese rango, en orden alfabético por sector.

- Falla si:
  - No se agregó al menos un mostrador en el aeropuerto y en ese caso el archivo de salida no debe crearse
- Ejemplo de invocación: Consultar los mostradores de todos los sectores indicando que en el sector A sólo hay un mostrador y está aceptando pasajeros del vuelo AC989 de AirCanada y no hay personas esperando en la cola. En el sector C hay 5 mostradores donde los mostradores 2 y 3 se usan para 3 vuelos de AmericanAirlines y hay 6 personas esperando en la cola de ese rango. Los demás mostradores están libres.

```
$> sh queryClient.sh -DserverAddress=10.6.0.1:50051
-Daction=queryCounters -DoutPath=../query1.txt
$> cat ../query1.txt
Sector  Counters  Airline          Flights             People
################################################################
A       (1-1)     AirCanada        AC989               0
C       (2-3)     AmericanAirlines AA123|AA124|AA125   6
C       (4-4)     -                -                   -
C       (7-8)     -                -                   -
D       (5-6)     -                -                   -
```

Para filtrar el resultado y sólo listar los mostradores del sector C

```
$> sh queryClient.sh -DserverAddress=10.6.0.1:50051
-Daction=queryCounters -DoutPath=../query1.txt -Dsector=C
$> cat ../query1.txt
Sector  Counters  Airline          Flights             People
################################################################
C       (2-3)     AmericanAirlines AA123|AA124|AA125   6
C       (4-4)     -                -                   -
C       (7-8)     -                -                   -
```

Para filtrar el resultado y sólo listar los mostradores de un sector Z inexistente o existente pero sin mostradores

```
$> sh queryClient.sh -DserverAddress=10.6.0.1:50051
-Daction=queryCounters -DoutPath=../query1.txt -Dsector=Z
$> cat ../query1.txt
Sector  Counters  Airline          Flights             People
################################################################
```

5.2 Consultar los check-ins realizados, filtrando por sector y aerolínea

- Funcionalidad: Consultar el historial de check-ins realizados en todos los mostradores, respetando el orden en que se realizaron, pudiendo listar únicamente los mostradores del sector `sectorName` con un parámetro opcional y/o los check-ins de pasajeros de vuelos de la aerolínea `airlineName` con un parámetro opcional, indicando para cada check-in el

sector, el mostrador donde se realizó el check-in, el nombre de la aerolínea, el código de vuelo y el código de la reserva

- Falla si:
  - No se realizó ningún check-in hasta el momento y en ese caso el archivo de salida no debe crearse
- Ejemplo de invocación: Consultar todos los check-ins realizados hasta el momento, respetando el orden en que se realizaron, donde se indica que se realizaron 3 check-ins considerando todos los mostradores de todos los sectores, donde el pasajero con el código de reserva ABC123 realizó el check-in en el mostrador 2 del Sector C para el vuelo AA123 de American Airlines y ese check-in se realizó antes de los demás check-ins de la salida

```
$> sh queryClient.sh -DserverAddress=10.6.0.1:50051 -Daction=checkins
-DoutPath=../query2.txt
$> cat ../query2.txt
   Sector  Counter   Airline           Flight     Booking
   ###################################################################
   C       2         AmericanAirlines  AA123      ABC321
   C       3         AmericanAirlines  AA123      XYZ999
   A       1         AirCanada         AC989      XYZ123
```

Para filtrar el resultado y sólo listar los mostradores del sector C

```
$> sh queryClient.sh -DserverAddress=10.6.0.1:50051 -Daction=checkins
-DoutPath=../query2.txt -Dsector=C
$> cat ../query2.txt
   Sector  Counter   Airline           Flight     Booking
   ###################################################################
   C       2         AmericanAirlines  AA123      ABC321
   C       3         AmericanAirlines  AA123      XYZ999
```

Para filtrar el resultado y sólo listar los check-ins de vuelos de AirCanada

```
$> sh queryClient.sh -DserverAddress=10.6.0.1:50051 -Daction=checkins
-DoutPath=../query2.txt -Dairline=AirCanada
   Sector  Counter   Airline           Flight     Booking
   ###################################################################
   A       1         AirCanada         AC989      XYZ123
```

Para filtrar el resultado y sólo listar los mostradores del sector C que realizaron check-ins de vuelos de AirCanada la salida es vacía

```
$> sh queryClient.sh -DserverAddress=10.6.0.1:50051 -Daction=checkins
-DoutPath=../query2.txt -Dairline=AirCanada
   Sector  Counter   Airline           Flight     Booking
   ###################################################################
```

5.3 Consultar el historial de un mostrador (***Exclusivo para el grupo de 5 integrantes***)

- Funcionalidad: Consultar el historial de asignaciones que involucran al mostrador `counterVal` del sector `sectorName`, respetando el orden que se realizaron las asignaciones, indicando para cada asignación, el rango de mostradores del cual formó parte, el nombre de la aerolínea correspondiente, los vuelos que aceptó y la cantidad total de check-ins realizados únicamente en ese mostrador durante la asignación.
- Falla si:
  - No existe un mostrador con ese número en ese sector
- Ejemplo de invocación: Consultar el historial de asignaciones que involucran al mostrador 7 del Sector C que se usó primero para 3 vuelos de AmericanAirlines donde el mostrador realizó 4 check-ins y perteneció a un rango de 3 mostradores que inició en el mostrador 6.

```
$> sh queryClient.sh -DserverAddress=10.6.0.1:50051 -Daction=history
-DoutPath=../query3.txt -Dsector=C -Dcounter=7
$> cat ../query3.txt
   Counters  Airline          Flights             Check-ins
   ########################################################################
   (6-8)     AmericanAirlines AA123|AA124|AA125   4
   (5-7)     AirCanada        AC987|AC101         0
```

Hechos y Consideraciones

Para simplificar el desarrollo se pueden tomar los siguientes considerandos como ciertos:

- No es necesario que tenga persistencia. Al reiniciar el sistema se comienza de cero la operación del mismo
- Se asume que el formato y contenido de los archivos de entrada es correcto y no es necesario validarlo
- Si el archivo de salida existe, reemplazar el contenido (no *appendear*)
- Los códigos de reserva no se repiten (incluso para vuelos y/o aerolíneas distintas)
- El nombre del sector no contiene espacios
- El nombre de la aerolínea no contiene espacios

Requisitos

Se requiere implementar:

- Todos los servicios deben ser implementados utilizando gRPC, teniendo en cuenta que los servicios deben poder atender pedidos de clientes de manera concurrente y responder a lo indicado en este enunciado
- Los clientes indicados cada uno como una aplicación diferente

Muy Importante:

➔ **Respetar exactamente los nombres de los *scripts*, los nombres de los archivos de salida y el orden y formato de los parámetros del *scripts***

➔ En todos los `pom.xml` que entreguen deberán definir el `artifactId` de acuerdo a la siguiente convención: "`tpe1-gX-Z`" donde X es el número de grupo y Z es `parent`, `api`, `server` o `client`. Por ejemplo: **`<artifactId>`**`tpe1-g7-api`**`</artifactId>`**

➔ En todos los `pom.xml` que entreguen deberán incluir el tag `name` con la siguiente convención: "`tpe1-gX-Z`" donde X es el número de grupo y Z es `parent`, `api`, `server` o `client`. Por ejemplo: **`<name>`**`tpe1-g7-api`**`</name>`**

➔ La implementación debe **respetar exactamente el formato de salida enunciado**

Material a entregar

Cada grupo deberá subir al **Campus ITBA** un archivo compactado conteniendo:

- El **código fuente** de la aplicación:
  Utilizando el arquetipo de Maven utilizado en las clases
  - Con una correcta separación de las clases en los módulos *api*, *client* y *server*
  - Un README indicando cómo preparar el entorno a partir del código fuente para correr el servidor y los cuatro clientes
  - El directorio oculto `.git/` donde se encuentra la historia de commits y modificaciones
  - **No se deben entregar los binarios**
- Un **documento breve** (no más de cuatro carillas) explicando:
  - Decisiones de diseño e implementación de los servicios
  - Criterios aplicados para el trabajo concurrente
  - Potenciales puntos de mejora y/o expansión

Corrección

El trabajo no se considerará aprobado si:

- No se entregó el trabajo práctico en tiempo y forma
- No se entrega alguno de los materiales solicitados en la sección anterior
- El código no compila utilizando Maven en consola (de acuerdo a lo especificado en el README a entregar)
- El servicio no inicia cuando se siguen los pasos del README
- Los clientes no corren al seguir los pasos del README

Si nada de esto se cumple, se procederá a la corrección donde se tomará en cuenta:

- Que los servicios y clientes funcionen correctamente según las especificaciones dadas
- El resultado de las pruebas y lo discutido en el coloquio
- La aplicación de los temas vistos en clase: Concurrencia y gRPC
- La modularización, diseño, testeo y reutilización de código
- El contenido y desarrollo del informe

Uso de Git

Es obligatorio el uso de un repositorio Git para la resolución de este TPE. Deberán crear un repositorio donde todos los integrantes del grupo colaboren con la implementación. No se aceptarán entregas que utilicen un repositorio git con un único *commit* que consista en la totalidad del código a entregar.

Los distintos *commits* deben permitir ver la evolución del trabajo, tanto grupal como individual.

Muy importante: **los repositorios creados deben ser privados, solo visibles para los integrantes del grupo y la cátedra en caso de que se lo solicite específicamente.**

Cronograma

- **Presentación del Enunciado: miércoles 10/04**
- **Entrega del trabajo: Estará disponible hasta el jueves 25/04 a las 23:59** la actividad "Entrega TPE 1" localizada en la sección Contenido / Evaluaciones / TPE 1. En la misma deberán cargar el paquete con el **código fuente** y el **documento**
- **El día miércoles 08/05 a las 18:00** cada grupo tendrá un espacio de 15 minutos para un coloquio. Durante el mismo se les hará una devolución del trabajo, indicando los principales errores cometidos. A criterio de la cátedra también se podrán realizar preguntas sobre la implementación. Opcionalmente se les podrá solicitar la ejecución de la aplicación a la cátedra para revisar el funcionamiento de alguna funcionalidad. Para ello es necesario que un integrante del equipo tenga el sistema remoto “levantado”.
- **El día del recuperatorio será el miércoles 26/06**
- **No se aceptarán entregas pasado el día y horario establecido como límite**

Dudas sobre el TPE

Las mismas deben volcarse en los **Debates** del Campus ITBA.

Recomendaciones

- **Clases y métodos útiles para consultar**
  - `java.nio.file.Files.lines`
  - `java.nio.file.Files.write`

Índice