

Trabajo Práctico Especial 1

Estación de Tren

16 de Abril de 2025



Créditos Foto: [@oxganggreen](#)

Objetivo

Implementar en grupos un **sistema remoto thread-safe** para la **gestión de trenes de pasajeros** que arriban a una estación ferroviaria, con notificaciones en tiempo real y ofreciendo reportes de la carga y descarga de pasajeros hasta el momento.

Introducción

En una **estación de tren** arriban **trenes de pasajeros**. Según el tamaño (cantidad de coches) existen **tres tipos: cortos, medianos y largos**. Según la configuración de **locomotoras** existen dos tipos: **tracción simple** (una locomotora) y **doble tracción** (dos locomotoras).

Los conductores de los trenes deben avisar que arribarán a la estación de forma que **el administrador de la estación le indica el andén al que debe dirigirse para realizar allí la descarga de pasajeros**. Luego cuando el conductor de tren lo indique **se realiza la descarga de pasajeros y luego la carga de otros pasajeros para retirarse del andén** (y por ende de la estación). Es decir, un andén se ocupa cuando un tren llega para realizar la descarga de pasajeros y se libera cuando se realiza la carga de pasajeros.

72.42 Programación de Objetos Distribuidos

La estación cuenta con una **cantidad limitada de andenes** y existen **tres tipos de andenes** (correspondientes a los tres tipos de trenes según el tamaño). Un tren puede ocupar un andén de su mismo tamaño o mayor (un tren mediano -M- no puede ocupar un andén corto -S-).

El tren con doble tracción tiene la posibilidad de separarse en dos trenes de la mitad del tamaño (*split*) para así ocupar dos andenes de la estación y realizar en ambas la descarga y carga de pasajeros. Al retirarse de los andenes los dos trenes vuelven a unirse en uno solo (*join*) y abandonan la estación. Por ejemplo, a un tren mediano de doble tracción se le puede asignar no sólo un andén mediano o un andén largo sino también dos andenes cortos. El proceso de separación en dos formaciones y luego la unión de las mismas es un proceso que sólo debe realizarse en el caso de no contar con un andén libre para la descarga pero si dos andenes libres de exactamente un tamaño menor al del tren (por ejemplo no cuenta con un andén libre mediano pero si existen dos andenes libres cortos). Los trenes con doble tracción de tamaño S no pueden dividirse.

En todo momento debe respetarse el orden de llegada a la estación. Un tren puede solicitar un andén y el administrador de la estación puede **indicarle uno de inmediato** (si es que hay uno libre que corresponda) **o esperar un tiempo hasta que algún andén adecuado se libere**. Para simplificar la implementación asumir que, una vez realizada la solicitud del andén, todos los trenes están esperando en un solo riel que llega a la estación.

Un andén puede cerrarse temporalmente (por ejemplo por mantenimiento o falta de personal) **cuando está vacío**, de forma que el mismo no podrá ser utilizado por los trenes que están esperando hasta que el mismo se vuelva a abrir. En el caso de que el administrador de la estación ya le haya indicado al conductor del tren a qué andén debe dirigirse pero el andén se cierra antes de que el conductor de tren haya decidido ocupar el andén para la descarga, el conductor del tren debe consultar la solicitud que realizó para ver si ya tiene otro andén al cuál dirigirse o debe esperar a que otro andén se desocupe.

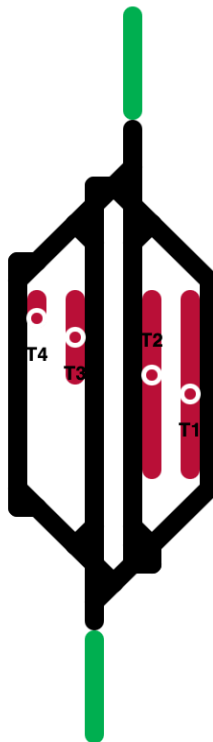


Diagrama de una estación con dos andenes para trenes largos (T1 y T2), un andén para trenes medianos (T3) y un andén para trenes cortos (T4).

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, **la implementación debe respetar la correcta separación entre servicio y cliente**. Para ello deben analizar si la lógica a implementar corresponde que esté del lado del servicio, del cliente o de ambas. Aunque en este trabajo práctico especial deberán implementar un único cliente de línea de comandos para cada uno de los servicios presentados considerar que un mismo servicio podría ser invocado por varios clientes implementados en diversas tecnologías (por ejemplo un cliente de línea de comandos y un cliente de una aplicación móvil).

1. Servicio de Administración de Andenes

- **Funcionalidad:** Agregar andenes, consultarlos y cambiar su estado
- **Usuario:** Administración de la estación
- **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 de andenes **platform.sh** y el resultado se debe imprimir en pantalla.


```
$> sh platform.sh -DserverAddress=xx.xx.xx.xx:yyyy -Daction=actionName [-Dsize=sizeType -Dplatform=platformNumber ]
```

donde

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

1.1 Agregar un andén

- **Funcionalidad:** Agregar un andén **a partir de su tamaño** (S, M o L). **Se indica el número del andén** agregado. El andén inicia abierto y libre. La numeración de los andenes es incremental donde el primero tiene el número 1.
- **No falla**
- **Ejemplo de cliente de invocación:** Agregar el andén “1” de tamaño medio “M”

```
$> sh platform.sh -DserverAddress=10.6.0.1:50051 -Daction=add -Dsize=M  
 Platform #1 (M) added
```


1.2 Estado actual de un andén

- **Funcionalidad:** Consultar el estado de un andén **a partir del número**. **Se indica si el andén está cerrado** (CLOSED), **abierto y libre** (IDLE), o **si está abierto y ocupado**. También se indica el número de andén provisto y **el tamaño** del mismo. En caso de que




72.42 Programación de Objetos Distribuidos

esté abierto y ocupado **se indica además el código del tren, su tamaño y la cantidad de pasajeros** que desembarcaron.

- Falla si:
 - No existe un andén con ese número
- Ejemplo de cliente de invocación: Consultar el estado del andén “1” que está abierto y libre


```
$> sh platform.sh -DserverAddress=10.6.0.1:50051 -Daction=status  
-Dplatform=1  
 Platform #1 (M) is IDLE
```

Al consultar el estado del andén “2” de tamaño corto que está abierto y ocupado con un tren que realizó la descarga de 10 pasajeros

```
$> sh platform.sh -DserverAddress=10.6.0.1:50051 -Daction=status  
-Dplatform=2  
 ABC123 (S) unloaded 10  in  Platform #2 (S)
```

1.3 Cambiar el estado de un andén

- Funcionalidad: Cambiar el estado de un andén de abierto (OPEN) a cerrado (CLOSED) o de cerrado a abierto (sólo si este está libre) **a partir del número. Se indica la misma información que 1.2.**
- Falla si:
 - No existe un andén con ese número
 - El andén está ocupado (no se pueden cerrar andenes ocupados)
- Ejemplo de cliente de invocación: Cambiar el estado del andén “1” que está abierto y libre y ahora estará cerrado y libre.

```
$> sh platform.sh -DserverAddress=10.6.0.1:50051 -Daction=toggle  
-Dplatform=1  
 Platform #1 (M) is CLOSED
```

2. Servicio de Gestión de Trenes

- Funcionalidad: Solicitar y ocupar un andén para la descarga de pasajeros y abandonar un andén luego de la carga de pasajeros
- Usuario: Conductor/a de tren
- 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 gestión de trenes **train.sh** y el resultado se debe imprimir en pantalla.

```
$> sh train.sh -DserverAddress=xx.xx.xx.xx:yyyy -Daction=actionName [  
-Did=trainCode -Dsize=sizeType -Dplatform=platformNumber  
-Doccupancy=occupancyNumber -Dtraction=double ]
```

donde

72.42 Programación de Objetos Distribuidos


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

2.1 Solicitar un andén o consultar el estado de una solicitud


- Funcionalidad: Solicitar un andén para la descarga de pasajeros **a partir del código del tren, su tamaño** (S, M o L), **la cantidad de pasajeros a bordo** y **su tracción** (simple o double). Se indica:
 - **El andén a donde debe dirigirse** para la descarga si es que había uno abierto y libre adecuado y el tren es el primero esperando
 - **Que está primero esperando pero aún no hay un andén** abierto y libre
 - **La cantidad de trenes que están esperando delante de él**

Para todos los casos se indica además los valores provistos de código, tamaño y cantidad de pasajeros. En caso de ya haber realizado una solicitud para un tren que todavía no abandonó la estación (desde la segunda invocación de 2.1 con el mismo código) se informa el estado de la misma y sólo es necesario indicar el código.

- Algoritmo determinístico para indicarle al tren que está primero esperando a qué andén debe dirigirse:
 - Priorizar que los trenes ocupen andenes de su tamaño (si un tren M solicita andén y están disponibles un andén tipo M y otro andén tipo L entonces elegir el andén de tipo M)
 - Priorizar los andenes más antiguos (si un tren solicita andén y están disponibles dos andenes #3 y #5 elegir el andén #3)
 - Sólo en caso de no conseguir un andén según los criterios anteriores, y exclusivamente para un tren de doble tracción, repetir el proceso anterior para las dos mitades del tren (*split*)
- Falla si:
 - Ya se invocó a 2.3 con el mismo código del tren (un tren que ya abandonó la estación no puede volver)
 - Ya se invocó a 2.2 con el mismo código del tren (un tren que ya ocupó un andén para la descarga de pasajeros no puede solicitar otro andén)
- Ejemplo de cliente de invocación: El tren con el código ABC123 de tamaño medio “M” con 10 pasajeros a bordo solicita un andén, donde si no se indica la tracción, se asume por defecto que es simple y se indica que hay 2 trenes esperando delante de él



```
$> sh train.sh -DserverAddress=10.6.0.1:50051 -Daction=request  
-Did=ABC123 -Dsize=S -Doccupancy=10  
 ABC123 (S) (10 🧑 ) is waiting for a platform with 2 trains ahead
```

Si más tarde se quiere conocer el estado de la solicitud para ese código no son necesarios los demás parámetros




```
$> sh train.sh -DserverAddress=10.6.0.1:50051 -Daction=request  
-Did=ABC123  
 ABC123 (S) (10 🧑 ) is waiting for a platform with 1 trains ahead
```



Si está primero esperando pero aún no hay un andén abierto y libre adecuado



72.42 Programación de Objetos Distribuidos

```
$> sh train.sh -DserverAddress=10.6.0.1:50051 -Daction=request  
-Did=ABC123  
 ABC123 (S) (10 ) is waiting for a platform with 0 trains ahead
```





Si es el primero esperando en riel y ya hay un andén abierto y libre adecuado indica a qué andén debe dirigirse (*proceed*) para más tarde poder ocuparlo y realizar la descarga de pasajeros (con 2.2)

```
$> sh train.sh -DserverAddress=10.6.0.1:50051 -Daction=request  
-Did=ABC123  
 ABC123 (S) (10 ) proceed to  Platform #2 (S)
```

El tren con el código DEF456 de tamaño medio “M” con 30 pasajeros a bordo solicita un andén, donde se indica que es de doble tracción con un parámetro opcional (y en la salida se indica como  en vez de )




```
$> sh train.sh -DserverAddress=10.6.0.1:50051 -Daction=request  
-Did=DEF456 -Dsize=M -Doccupancy=30 -Dtraction=double  
 DEF456 (M) (30 ) is waiting for a platform with 3 trains ahead
```

Si el tren de doble tracción es el primero esperando en riel y corresponde la separación en dos formaciones se indican los dos andenes a los que debe dirigirse (*split and proceed*).

```
$> sh train.sh -DserverAddress=10.6.0.1:50051 -Daction=request  
-Did=DEF456  
 DEF456 (M) (30 ) split and proceed to  Platform #2 (S) and   
Platform #3 (S)
```

2.2 Ocupar un andén y realizar la descarga de pasajeros

- Funcionalidad: Ocupar un andén con el tren que está primero esperando en el riel, según la indicación de “*proceed*” o “*split and proceed*” de 2.1, **a partir del número de andén y el código del tren**. Se indica el **número de andén** provisto y su **tamaño**, el **código de tren** provisto, su **tamaño** y la **cantidad de pasajeros** que desembarcaron.
- Falla si:
 - No existe un andén con ese número
 - El tren no es el primero esperando en el riel (como es una sola vía hay que respetar el orden en que los trenes invocaron a 2.1)
 - El andén no corresponde con la indicación de “*proceed*” o “*split and proceed*” de 2.1
- Ejemplo de cliente de invocación: Ocupar el andén “2” con el tren de código ABC123

```
$> sh train.sh -DserverAddress=10.6.0.1:50051 -Daction=proceed  
-Did=ABC123 -Dplatform=2  
 ABC123 (S) unloaded 10  in  Platform #2 (S)
```

72.42 Programación de Objetos Distribuidos

Si 2.1 indicó “*split and proceed*” se deben realizar dos invocaciones y sólo la segunda realiza la descarga de todos los pasajeros. Suponiendo un tren de código DEF456 con 30 pasajeros:

```
$> sh train.sh -DserverAddress=10.6.0.1:50051 -Daction=proceed  
-Did=DEF456 -Dplatform=2  
🚆 DEF456 (M) unloaded 0 🧑 in 🚆 Platform #2 (S)  
$> sh train.sh -DserverAddress=10.6.0.1:50051 -Daction=proceed  
-Did=DEF456 -Dplatform=3  
🚆 DEF456 (M) unloaded 30 🧑 in 🚆 Platform #3 (S)
```

2.3 Realizar la carga de pasajeros y liberar el andén

- Funcionalidad: Liberar un andén que se ocupó con 2.2 a **partir del número de andén, el código del tren y la cantidad de pasajeros que embarcaron** en el tren antes de abandonar la estación. Se indica el **número de andén** provisto y su **tamaño**, el **código de tren** provisto, su **tamaño** y la **cantidad de pasajeros** provista.
- Falla si:
 - No existe un andén con ese número
 - El andén está cerrado o libre
- Ejemplo de cliente de invocación: El tren con el código ABC123 realiza la carga de 3 pasajeros y abandona la estación

```
$> sh train.sh -DserverAddress=10.6.0.1:50051 -Daction=depart -Did=ABC123  
-Dplatform=2 -Doccupancy=3  
🚆 ABC123 (S) left 🚆 Platform #2 (S) after loading 3 🧑
```

Si 2.1 indicó *split and proceed* se deben realizar dos invocaciones y sólo la segunda realiza la carga de todos los pasajeros y libera el andén (donde se debe contabilizar la suma de los pasajeros de ambas invocaciones)

```
$> sh train.sh -DserverAddress=10.6.0.1:50051 -Daction=depart -Did=DEF456  
-Dplatform=2 -Doccupancy=12  
🚆 DEF456 (M) ready to leave 🚆 Platform #2 (S)  
$> sh train.sh -DserverAddress=10.6.0.1:50051 -Daction=depart -Did=DEF456  
-Dplatform=3 -Doccupancy=14  
🚆 DEF456 (M) rejoined and left 🚆 Platform #3 (S) after loading
```

26 🧑

3. Servicio de Tablero de Arribos y Partidas

- Funcionalidad: Consultar el estado actual del tablero de arribos y partidas, recibir la versión actualizada del tablero en tiempo real y administrar los anuncios en los andenes.
- Usuario: Pasajeros de la estación y personal de plataforma de la estación
- 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 tablero de arribos y partidas de trenes **train.sh** y el resultado se debe imprimir en pantalla.

72.42 Programación de Objetos Distribuidos

```
$> sh board.sh -DserverAddress=xx.xx.xx.xx:yyyy -Daction=actionName
```






donde

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

3.1 Estado actual del tablero

- Funcionalidad: Consultar el **estado actual del tablero** indicando, para cada andén de la estación:
 - Número
 - Tamaño
 - Estado
 - i. Cerrado (CLOSED)
 - ii. Abierto y libre (IDLE)
 - iii. Abierto y ocupado, indicando el código del tren y su tamaño
- No falla
- Ejemplo de cliente de invocación: Consultar el estado actual del tablero donde existen cinco andenes donde el tren de tracción simple de código ABC123 realizó la descarga de pasajeros en el andén 2 y el tren de tracción doble de código DEF456 realizó la descarga de pasajeros en los andenes 4 y 5 (según la indicación de un *split and proceed*).

```
$> sh board.sh -DserverAddress=10.6.0.1:50051 -Daction=snapshot
```

Platform	Size	Status
1	M	IDLE
2	M	 ABC123 (S)
3	L	CLOSED
4	S	  DEF456 (M)
5	S	  DEF456 (M)

3.2 Actualización en tiempo real del tablero y anuncios en andenes

- Funcionalidad:
 - **Consultar en tiempo real el tablero** indicando, para cada andén de la estación, la misma información que 3.1 donde el cliente no termina y ante cualquier evento en el sistema que implique un cambio en el tablero éste se vuelve a mostrar por completo con la información actualizada.
 - Publicar anuncios (destinados a los pasajeros) que se asocian a los andenes y que se consultan únicamente en este método. Un anuncio consiste en un mensaje corto (String) que se visualizará en la consulta en tiempo real del tablero. Todos los andenes al agregarse inician con un anuncio vacío "". Al publicar un anuncio en un andén que ya tiene uno, se reemplaza. Eliminar un anuncio de un andén consiste en publicar uno vacío.
- Falla si:
 - No existe un andén con el número provisto al momento de publicar un anuncio

72.42 Programación de Objetos Distribuidos

- Ejemplo de cliente de invocación: Consultar en tiempo real el tablero donde los andenes tienen el anuncio vacío por defecto y el cliente no termina

```
$> sh board.sh -DserverAddress=10.6.0.1:50051 -Daction=live
### LIVE BOARD ###
Platform | Size | Status
1        | M   | IDLE
2        | M   | 🚆ABC123 (S)
3        | L   | CLOSED
4        | M   | 🚆DEF456 (M)
###
Number and Announcement: _
```

Si por ejemplo ocurre el evento de carga de pasajeros y el andén 4 se libera (por una invocación a 2.3) como esto implica cambios en el tablero éste se vuelve a mostrar a continuación

```
$> sh board.sh -DserverAddress=10.6.0.1:50051 -Daction=live
### LIVE BOARD ###
Platform | Size | Status
1        | M   | IDLE
2        | M   | 🚆ABC123 (S)
3        | L   | CLOSED
4        | M   | 🚆DEF456 (M)
###
🚆 Number and Announcement: _
### LIVE BOARD ###
Platform | Size | Status
1        | M   | IDLE
2        | M   | 🚆ABC123 (S)
3        | L   | CLOSED
4        | M   | IDLE
###
🚆 Number and Announcement: _
```

En el renglón de 🚆 **Number and Announcement:** el cliente debe leer de entrada estándar el número de andén y la cadena de caracteres del anuncio a publicar. Si por ejemplo el usuario decide interactuar con la terminal y publicar el anuncio “Leaving Soon” para el andén 2 debe escribir “2 Leaving Soon” y el anuncio queda publicado, siendo otro evento del sistema para actualizar en tiempo real el tablero.

```
$> sh board.sh -DserverAddress=10.6.0.1:50051 -Daction=live
### LIVE BOARD ###
Platform | Size | Status
1        | M   | IDLE
2        | M   | 🚆ABC123 (S)
3        | L   | CLOSED
4        | M   | 🚆DEF456 (M)
###
🚆 Number and Announcement: _
```

72.42 Programación de Objetos Distribuidos

```
### LIVE BOARD ###
Platform | Size | Status
1         | M   | IDLE
2         | M   | 🚆ABC123 (S)
3         | L   | CLOSED
4         | M   | IDLE
###
🚆 Number and Announcement: 2 Leaving Soon
### LIVE BOARD ###
Platform | Size | Status
1         | M   | IDLE
2         | M   | 🚆ABC123 (S) 🚦 Leaving Soon
3         | L   | CLOSED
4         | M   | IDLE
###
🚆 Number and Announcement: _
```

Para simplificar la implementación del cliente, si mientras el usuario está escribiendo un anuncio ocurre un evento que dispara una actualización del tablero es esperable que se pierda el texto ingresado sin confirmar. Para finalizar la ejecución del cliente basta con ejecutar la combinación de teclas para terminar un comando de la terminal (^C).

3.3 Eliminar todos los anuncios (Exclusivo para grupos de 4 integrantes)

- Funcionalidad: Eliminar, para cada andén de la estación, el anuncio (esto es, publica el anuncio vacío). Se indica cuántos anuncios se eliminaron (anuncios no vacíos que se reemplazaron por uno vacío).
- No falla
- Ejemplo de cliente de invocación: Eliminar los anuncios de todos los andenes donde sólo un andén tenía un anuncio definido (no vacío)

```
$> sh board.sh -DserverAddress=10.6.0.1:50051 -Daction=purge
1 announcements were removed
```

4. Servicio de Notificación de Andenes

- Funcionalidad: Registrarse para recibir en tiempo real notificaciones de eventos relacionados a un andén o a todos los andenes de la estación y anular el registro para dejar de recibir notificaciones
- Usuario: Personal de plataforma de la estación
- 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 notificación de andenes **alert.sh** y el resultado se debe imprimir en pantalla.

```
$> sh alert.sh -DserverAddress=xx.xx.xx.xx:yyyy -Daction=actionName [
-Dplatform=platformNumber ]
```

donde

72.42 Programación de Objetos Distribuidos

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


4.1 Registrar a un andén para recibir notificaciones

- Funcionalidad: Registrar a un andén **a partir de su número** para que sea notificado en tiempo real de todos los eventos relacionados a ese andén.
- Se considera un evento:

- Se **registró correctamente** para recibir notificaciones (evento inicial)

 Platform #1 registered for alerts




- Se **cambió el estado** del andén (invocando a 1.3)

 Platform #1 (M) is CLOSED

- Se indicó que un **tren debe dirigirse al andén** (según el “*proceed*” o “*split and proceed*” de 2.1)

 ABC123 (S) (10 ) proceed to  Platform #2 (S)



- Se **realizó la descarga de pasajeros** ocupando el andén (invocando a 2.2)

 ABC123 (S) unloaded 10  in  Platform #2 (S)

- Se **realizó la carga de pasajeros** liberando el andén (invocando a 2.3)

 ABC123 (S) left  Platform #2 (S) after loading 3 

- Se **publicó un anuncio** para el andén (invocando a 3.2 e interactuando con la consola)


 Platform #2 (M)  Delayed

- Se **eliminó un anuncio** para el andén (invocando a 3.3)

 Platform #2 (M) 




- Falla si:
 - No existe un andén con ese número
 - El andén ya se registró para recibir notificaciones con 4.1 y no se invocó a 4.2 con ese número de andén
- Ejemplo de cliente de invocación: Registrar al andén de número 1 para recibir notificaciones, mostrando el evento inicial, donde el cliente no termina y muestra las notificaciones a medida que van ocurriendo

72.42 Programación de Objetos Distribuidos

```
$> sh alert.sh -DserverAddress=10.6.0.1:50051 -Daction=register  
-Dplatform=1  
 Platform #1 registered for alerts  
...
```

4.2 Anular el registro de un andén

- Funcionalidad: Anular el registro de un andén para dejar de recibir más notificaciones. Además finaliza la ejecución del cliente de notificaciones correspondiente (de 4.1).
- Falla si:
 - El andén no se registró para recibir notificaciones (no se invocó a 4.1 con ese número de andén)
- Ejemplo de invocación: Desde la consola t2 anular el registro del andén de número “1” para dejar de recibir notificaciones en la consola t1 (donde se invocó a 4.1), mostrando el evento final de anulación de registro tanto en la consola t1 como en la t2, y finalizando ambos clientes:


<pre>t1\$> sh alert.sh -DserverAddress=10.6.0.1:50051 -Daction=register -Dplatform=1  Platform #1 registered for alerts ...  Platform #1 unregistered t1\$> _</pre>	<pre>t2\$> sh alert.sh -DserverAddress=10.6.0.1:50051 -Daction=unregister -Dplatform=1  Platform #1 unregistered t2\$> _</pre>
--	--

4.3 Registrar a la estación para recibir notificaciones (Exclusivo para grupos de 4 integrantes)

- Funcionalidad: Registrar a la estación para que sea notificada en tiempo real de todos los eventos relacionados a todos los andenes (cubriendo tanto los andenes existentes al momento de la invocación del método como los nuevos andenes que se agreguen con 1.1).
- Se considera un evento:
 - Se **registró correctamente** para recibir notificaciones indicando la cantidad de andenes al momento de la invocación (evento inicial)

Station registered for alerts for 3 platforms

- Se **agregó un andén** (invocando a 1.1) indicando la salida de 1.1

 Platform #1 (M) added

- **Los eventos mencionados en 4.1** (a excepción del evento inicial)

- No falla (Se debe permitir por ejemplo que desde dos consolas distintas t1 y t2 se invoque a 4.3 y las notificaciones llegan en tiempo real en ambas consolas)

72.42 Programación de Objetos Distribuidos

- Ejemplo de cliente de invocación: Registrar a la estación con tres andenes para recibir notificaciones, mostrando el evento inicial, donde el cliente no termina y muestra las notificaciones a medida que van ocurriendo

```
$> sh alert.sh -DserverAddress=10.6.0.1:50051 -Daction=station
Station registered for alerts for 3 platforms
🚆ABC123 (S) (10 👤 ) proceed to 🚆 Platform #2 (S)
🚆 Platform #1 (M) 📶 Expect delays
🚆ABC123 (S) left 🚆 Platform #2 (S) after loading 3 👤
🚆 Platform #4 (L) added
...
```

5. Servicio de Reportes

- Funcionalidad: Consultar los trenes que están esperando un andén para la descarga, los trenes que realizaron la carga y abandonaron la estación y todos los eventos que ocurrieron en la estación
- Usuario: Ministerio de Transporte
- 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 reportes **data.sh** y el resultado se debe imprimir en pantalla.

```
$> sh reports.sh -DserverAddress=xx.xx.xx.xx:yyyy -Daction=actionName [
-DoutPath=filePath.txt | -Dplatform=platformNumber ]
```

donde





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

5.1 Trenes que solicitaron un andén y no descargaron pasajeros

- Funcionalidad: Consultar los trenes que realizaron una solicitud de andén y no realizaron aún la descarga de pasajeros (no se invocó a 2.2 para esos trenes), indicando el código del tren, su tamaño y la cantidad de pasajeros a bordo. Respetar el orden en que solicitaron el andén, donde se listan primero los que lo solicitaron antes.
- Falla si:
 - No existe al menos un tren que solicitó un andén y todavía no realizó la descarga y en ese caso el cliente no debe crear el archivo de salida
- Ejemplo de invocación: Consultar los trenes que solicitaron un andén y no realizaron la descarga, dejando el resultado en un archivo `query1.txt`










```
$> sh reports.sh -DserverAddress=10.6.0.1:50051 -Daction=waiting
-DoutPath=/tmp/query1.txt
$> cat /tmp/query1.txt
🚆ABC123 (S) (10 👤 )
```

72.42 Programación de Objetos Distribuidos







```
 XYZ987 (M) (5 )  
 DEF456 (S) (15 )  
...
```

5.2 Trenes que cargaron pasajeros y abandonaron la estación

- Funcionalidad: Consultar los trenes que realizaron la carga de pasajeros (ya se invocó 2.3 para esos trenes) y abandonaron la estación, indicando el código del tren, su tamaño y la cantidad de pasajeros que embarcaron, además del número de andén liberado y su tamaño. Respetar el orden en que abandonaron la estación, donde se listan primero los que abandonaron antes. Se debe ofrecer también la posibilidad de filtrar el resultado listando únicamente los trenes liberaron un andén en particular con el número del andén como parámetro opcional.
- Falla si:
 - No existe al menos un tren que cargó pasajeros y abandonó la estación y en ese caso el cliente no debe crear el archivo de salida
- Ejemplo de invocación: Consultar los trenes que cargaron pasajeros y abandonaron la estación, dejando el resultado en un archivo query2.txt

```
$> sh reports.sh -DserverAddress=10.6.0.1:50051 -Daction=left  
-DoutPath=/tmp/query2.txt  
$> cat /tmp/query2.txt  
 ABC123 (S) left  Platform #2 (S) after loading 3   
 XYZ987 (M) left  Platform #1 (M) after loading 6   
 DEF456 (S) left  Platform #2 (S) after loading 3   
...
```

Para filtrar el resultado y sólo listar los trenes que abandonaron la estación liberando el andén "2":

```
$> sh reports.sh -DserverAddress=10.6.0.1:50051 -Daction=left  
-DoutPath=/tmp/query2.txt -Dplatform=2  
$> cat /tmp/query2.txt  
 ABC123 (S) left  Platform #2 (S) after loading 3   
 DEF456 (S) left  Platform #2 (S) after loading 3   
...
```

5.3 Notificaciones de la estación (Exclusivo para grupos de 4 integrantes)

- Funcionalidad: Listar todos los eventos relacionados a todos los andenes que sucedieron desde el inicio del servicio. Se debe ofrecer también la posibilidad de filtrar el resultado listando únicamente los eventos relacionados a un andén en particular con el número del andén como parámetro opcional.
- Falla si:
 - No se agregaron andenes (no se invocó a 1.1) y en ese caso el cliente no debe crear el archivo de salida
- Ejemplo de cliente de invocación: Listar todos los eventos de la estación, dejando el resultado en un archivo query3.txt

72.42 Programación de Objetos Distribuidos

```
$> sh reports.sh -DserverAddress=10.6.0.1:50051 -Daction=alerts  
-DoutPath=/tmp/query3.txt  
$> cat /tmp/query3.txt  
...  
🚆ABC123 (S) (10 👤 ) proceed to 🚆 Platform #2 (S)  
🚆 Platform #1 (M) 🚧 Expect delays  
🚆ABC123 (S) left 🚆 Platform #2 (S) after loading 3 👤  
🚆 Platform #4 (L) added  
...
```

Para filtrar el resultado y sólo listar las notificaciones relacionadas al andén “2”:

```
$> sh reports.sh -DserverAddress=10.6.0.1:50051 -Daction=alerts  
-DoutPath=/tmp/query3.txt -Dplatform=2  
$> cat /tmp/query3.txt  
...  
🚆ABC123 (S) (10 👤 ) proceed to 🚆 Platform #2 (S)  
🚆ABC123 (S) left 🚆 Platform #2 (S) after loading 3 👤  
...
```

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
- Los valores de los parámetros de los clientes no contienen espacios
- Si el archivo de salida existe, reemplazar el contenido (no *appendear*)

Requisitos

Se requiere implementar:

- Todos los servicios 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 de consola diferente

Muy Importante:

→ **Respetar EXACTAMENTE**

- ◆ **Los nombres de los *scripts* y sus parámetros** (Si se pide -Dplatform no se aceptará -Dplataforma, -Dbay, -Dtrack, etc.)
- ◆ **El orden y formato de salida enunciado, tanto para los clientes de consola como para los archivos TXT de salida**

→ **En TODOS los pom.xml que entreguen deberán definir**

72.42 Programación de Objetos Distribuidos

- ◆ 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>`
- ◆ El 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>`

Material a entregar

Cada grupo deberá subir al **Campus** **UNICAMENTE 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 cinco clientes
 - El directorio oculto `.git/` donde se encuentra la historia de commits y modificaciones. Recordar que la descarga desde github.com no incluye el directorio `.git/`.
 - **No se deben entregar los binarios.** Recordar de ejecutar el comando `mvn clean` antes de la entrega.
- 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 la 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 16/04**
- **Entrega del trabajo: Estará disponible hasta el sábado 03/05 a las 23:59** la actividad "TPE 1" localizada en la sección Contenido / Evaluaciones. En la misma deberán cargar el archivo compactado.
- **El día del coloquio será el miércoles 07/05.**
- **El día del recuperatorio será el miércoles 25/06**
- **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

- Para el parseo de argumentos del cliente como `-Dplatform=1` consultar [`java.lang.System#getProperty\(java.lang.String\)`](#)
- Para la escritura de archivos `.txt` consultar [`java.nio.file.Files.write`](#)