Trabajo Práctico Especial 1: Despegues

01 de Septiembre de 2021

Objetivo

Implementar en **grupos** un **sistema remoto *thread-safe*** para registrar los despegues de un aeropuerto, a partir de la existencia de una o más pistas de despegue (a partir de ahora pistas), permitiendo notificar a las aerolíneas de los eventos y ofreciendo reportes de los despegues hasta el momento.

Descripción Funcional

Servicios Remotos

El sistema requiere el desarrollo de **los siguientes servicios remotos**:

Servicio de Administración

- Funcionalidad: Administración de pistas y órdenes de despegue.
- Usuario: Operaciones del aeropuerto
- Debe contar con métodos para:
    - **Agregar una pista** a partir de un nombre y una categoría.
        i. Una pista se identifica a partir de su **nombre**, por lo que no pueden existir dos pistas con el mismo nombre. Si la pista a agregar ya existe, se debe arrojar un error.
        ii. Existen **seis categorías (fijas) de pistas**: A, B, C, D, E y F. La clasificación surge del código de referencia aeroportuario[1]. Esta categoría permitirá que ciertos vuelos puedan despegar sólo en pistas de cierta categoría (ver explicación más adelante).
        iii. Cada pista podrá estar abierta o cerrada. Sólo se permitirán despegues en pistas abiertas. Al agregar una pista, ésta se inicia abierta.
    - **Consultar el estado de una pista**: indicando si está abierta o cerrada. Si no existe una pista con ese nombre entonces debe arrojar un error.
    - **Abrir y cerrar una pista**. Si no existe una pista con ese nombre o si se ordena abrir una pista que ya estaba abierta, se debe arrojar un error (lo mismo con cerrada).
    - **Emitir una orden de despegue en las pistas:** Una orden de despegue consiste en el despegue de un vuelo en cada una de las pistas abiertas con vuelos esperando despegar.
    - **Emitir una orden de reordenamiento en las pistas:** Una orden de reordenamiento consiste en reubicar a todos los vuelos que están esperando despegar en todas las pistas (abiertas y cerradas). Se procede a liberar las pistas y encolar cada vuelo como si se hiciera una nueva solicitud de despegue. Para

[1] https://es.wikipedia.org/wiki/Código_de_referencia_aeroportuario

liberar se saca un vuelo por cada pista existente. El orden de selección de pista es por categoría y nombre ascendente.

Servicio de Solicitud de Pista

- Funcionalidad: Solicitar pista
- Usuario: Tripulación de un vuelo
- Debe contar con un método para:
    - **Emitir una solicitud de pista de un vuelo** a partir de un código identificador, un código del aeropuerto de destino, el nombre de la aerolínea y la categoría mínima requerida para despegar.
        i. Se garantiza que no existirán dos vuelos con el mismo código identificador.
        ii. Cada vuelo cuenta con una **categoría de pista mínima** requerida para poder realizar el despegue. Esto implica que:
            1. Un vuelo que requiere una categoría mínima A puede despegar en pistas abiertas de cualquier categoría.
            2. Un vuelo que requiere una categoría mínima E puede despegar únicamente en pistas abiertas de categoría E y F.
        iii. **El algoritmo de asignación de pista es el siguiente:** Dentro de las pistas abiertas y con las categorías requeridas se selecciona la que tenga menos vuelos esperando despegar. En caso de que exista más de una se desempata por categoría de pista mínima y nombre (ambos alfabéticamente ascendente). En caso de no encontrar una pista se debe lanzar un error.

Servicio de Seguimiento de Vuelo

- Funcionalidad: Registro de una aerolínea para ser notificado de los eventos de una solicitud de pista al momento que ocurren.
- Usuario: Personal de la aerolínea de un vuelo
- Debe contar con un método para:
    - Registrar a una aerolínea para que ésta sea notificada de los eventos relacionados a la solicitud de pista de un vuelo a partir del código identificador del vuelo.
        - Se considera un evento:
            - El vuelo fue asignado a una pista
            - El vuelo cambió su posición en la cola de espera de la pista
            - El vuelo despegó
        - En los dos primeros se informa el código identificador del vuelo, la pista en la cual está esperando y cuántos vuelos hay adelante en esa pista. En el tercero vuelo y pista.
        - Varios miembros de una aerolínea pueden registrarse para ser notificados de los eventos de una misma solicitud de pista.
        - Se debe arrojar error en caso de que:
            - Se quiera registrar para un vuelo que no pertenezca a la aerolínea
            - Si el vuelo no está esperando a despegar.

Servicio de Consulta

- Funcionalidad: Consultar los despegues
- Usuario: Administración del aeropuerto

72.42 Programación de Objetos Distribuidos

- Debe contar con un método para:
    - Consultar los despegues del aeropuerto hasta el momento, listando los vuelos que despegaron:
        i. En el aeropuerto
        ii. En una sola pista
        iii. De una sola aerolínea

Clientes

Para poder probar el sistema se requiere que se implementen cuatro programas cliente, cada uno coincidente con cada interfaz remota.

Cliente de Administración

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 y el resultado se debe imprimir en pantalla.

Por ejemplo:

```
$> ./run-management -DserverAddress=xx.xx.xx.xx:yyyy -Daction=actionName
[ -Drunway=runwayName | -Dcategory=minCategory ]
```

donde

- `run-management` es el script que invoca a la clase del cliente de administración.
- `xx.xx.xx.xx:yyyy` es la dirección IP y el puerto donde está publicado el servicio de administración de los despegues.
- `actionName` es el nombre de la acción a realizar.
    - `add`: Agrega una pista de categoría `minCategory` con el nombre `runwayName`. Deberá imprimir en pantalla el estado de la pista luego de agregarla o el error correspondiente.
    - `open`: Abre la pista `runwayName`. Deberá imprimir en pantalla el estado de la pista luego de invocar a la acción o el error correspondiente.
    - `close`: Cierra la pista `runwayName`. Deberá imprimir en pantalla el estado de la pista luego de invocar a la acción o el error correspondiente.
    - `status`: Consulta el estado de la pista `runwayName`. Deberá imprimir en pantalla el estado de la pista al momento de la consulta.
    - `takeOff`: Emite una orden de despegue en las pistas abiertas. Deberá imprimir en pantalla la finalización de la acción.
    - `reorder`: Emite una orden de reordenamiento en las pistas. Deberá imprimir en pantalla la cantidad de vuelos que obtuvieron una pista y detallar aquellos que no.

De esta forma,

```
$> ./run-management -DserverAddress=10.6.0.1:1099 -Daction=add
-Drunway=R1 -Dcategory=D
```

agrega al aeropuerto una pista de categoría `D` llamada `R1` utilizando el servicio remoto publicado en `10.6.0.1:1099` e imprime en pantalla el mensaje correspondiente:

```
Runway R1 is open.
```

En el caso de solicitar un reordenamiento invocando a

```
$> ./run-management -DserverAddress=10.6.0.1:1099 -Daction=reorder
```

y suponiendo un aeropuerto con cinco vuelos esperando despegar, de los cuales el vuelo de código identificador `345` no pudo solicitar pista (porque al momento del reordenamiento no existe una pista con la categoría mínima requerida para ese vuelo) se imprime:

```
Cannot assign Flight 345.
4 flights assigned.
```

Cliente de Solicitud de Pista

Deberá leer de un archivo CSV un lote de solicitudes de pista, donde el delimitador de campos es un punto y coma.

Detalle del archivo de entrada del cliente de solicitud de pista, a partir de ahora `vuelos.csv`

- **Campos:**
  - **El código identificador numérico**
  - **El código del aeropuerto de destino**
  - **El nombre de la aerolínea**
  - **La categoría de pista mínima requerida**

Cada línea del archivo representa una solicitud de pista de un vuelo conteniendo los datos de cada uno de los campos mencionados, separados por `;`. El archivo contendrá una primera línea de encabezado.

Ejemplo de las primeras cuatro líneas de archivo:

```
FlightCode;DestinyAirport;AirlineName;MinimumCategory
5382;SCL;Air Canada;F
926;COR;Aerolíneas Argentinas;C
927;COR;Aerolíneas Argentinas;C
```

- En la segunda línea se solicita pista para el vuelo con destino a SCL de la aerolínea Air Canada que requiere una pista de categoría F para despegar.
- En la tercera línea se solicita pista para el vuelo con destino a COR de la aerolínea Aerolíneas Argentinas que requiere una pista de categoría C o superior para despegar.

72.42 Programación de Objetos Distribuidos

El *path* del archivo `vuelos.csv`, necesario para emitir un lote de solicitudes de pista, se recibe a través de argumentos de línea de comando al llamar a la clase del cliente de solicitud de pista y el resultado se debe imprimir en pantalla.

Por ejemplo:

```
$>  ./run-runway -DserverAddress=xx.xx.xx.xx:yyyy -DinPath=fileName
```

donde

- `run-runway` es el script que invoca a la clase del cliente de solicitud de pista.
- `xx.xx.xx.xx:yyyy` es la dirección IP y el puerto donde está publicado el servicio de solicitud de pista.
- `fileName` es el path del archivo de entrada con las solicitudes de pista

De esta forma,

```
$> ./run-runway -DserverAddress=10.6.0.1:1099 -DinPath=../vuelos.csv
```

realiza la solicitud de pista de los vuelos presentes en el archivo `vuelos.csv` que se encuentra ubicado en el directorio superior a donde está el intérprete de comandos, utilizando el servicio remoto publicado en `10.6.0.1:1099` e imprime en pantalla un mensaje luego de haber realizado todas las solicitudes. Por ejemplo, para un archivo `vuelos.csv` con 100 registros se imprime:

```
100 flights assigned.
```

En el caso de que la solicitud de pista de un vuelo ocasione un error porque el aeropuerto no cuenta con al menos una pista abierta con la categoría mínima requerida, se deben imprimir el detalle de los vuelos que no pudieron solicitar pista. Para el CSV de ejemplo de cuatro líneas mencionado más arriba y con un aeropuerto que no cuenta con una pista abierta de categoría F se imprime:

```
Cannot assign Flight 5382.
2 flights assigned.
```

Cliente de Seguimiento de Vuelo

El nombre de la aerolínea y el código identificador del vuelo se recibe a través de argumentos de línea de comando al llamar al script del cliente de seguimiento de vuelo y el resultado se debe imprimir en pantalla.

Por ejemplo:

```
$> ./run-airline -DserverAddress=xx.xx.xx.xx:yyyy -Dairline=airlineName
-DflightCode=flightCode
```

donde

- `run-airline` es el script que invoca a la clase del cliente de seguimiento de vuelo.
- `xx.xx.xx.xx:yyyy` es la dirección IP y el puerto donde está publicado el servicio de seguimiento de vuelo.
- `airlineName`: el nombre de la aerolínea
- `flightCode`: el código identificador de un vuelo de la aerolínea `airlineName` que esté esperando despegar.

De esta forma,

```
$>  ./run-airline -DserverAddress=10.6.0.1:1099 -DflightCode=5382
-DairlineName='Air Canada'
```

registra a un agente de la aerolínea `Air Canada` para recibir notificaciones del vuelo `5382` con destino a SCL utilizando el servicio remoto publicado en `10.6.0.1:1099` e imprime en pantalla un mensaje al momento de la registración. Cada vez que se produce un evento relacionado al vuelo en cuestión se emite un mensaje. Una salida posible es la siguiente:

Flight 5382 with destiny SCL was assigned to runway R1 and there are 3 flights waiting ahead.

A flight departed from runway R1. Flight 5382 with destiny SCL has 2 flights waiting ahead.

Flight 5382 with destiny SCL was assigned to runway R3 and there are 0 flights waiting ahead.

Flight 5382 with destiny SCL departed on runway R3.

Cliente de Consulta

Deberá dejar en archivos CSV los resultados de las consultas realizadas. Las consultas posibles son las siguientes:

Consulta 1: Despegues

Cada línea de la salida debe contener separados por `;` los siguientes campos:

- la cantidad de órdenes de despegue que se emitieron desde la orden de solicitud de pista
- el nombre de la pista donde despegó
- el código identificador del vuelo
- el aeropuerto de destino
- el nombre de la aerolínea.

Se debe respetar el orden de despegue.

❏ Salida de ejemplo:

`TakeOffOrders;RunwayName;FlightCode;DestinyAirport;AirlineName`
`0;R1;926;COR;Aerolíneas Argentinas`

```
0;R2;757;BRC;Aerolíneas Argentinas
3;R3;5382;SCL;Air Canada
0;R1;927;COR;Aerolíneas Argentinas
```

Consulta 2: Despegues de los vuelos de una aerolínea

Respetando lo enunciado en la Consulta 1, se deben listar únicamente los vuelos de una aerolínea donde el nombre se recibe por parámetro.

- ❏ Salida de ejemplo para la aerolínea “Aerolíneas Argentinas”:

```
TakeOffOrders;RunwayName;FlightCode;DestinyAirport;AirlineName
0;R1;926;COR;Aerolíneas Argentinas
0;R2;757;BRC;Aerolíneas Argentinas
0;R1;927;COR;Aerolíneas Argentinas
```

Consulta 3: Despegues de los vuelos de una pista

Respetando lo enunciado en la Consulta 1, se deben listar únicamente los vuelos que hayan despegado en una pista donde el nombre se recibe por parámetro.

- ❏ Salida de ejemplo para la pista “R1”:

```
TakeOffOrders;RunwayName;FlightCode;DestinyAirport;AirlineName
0;R1;926;COR;Aerolíneas Argentinas
0;R1;927;COR;Aerolíneas Argentinas
```

La información de cuál es la consulta a correr y los parámetros necesarios se reciben a través de argumentos de línea de comando al llamar a la clase del cliente de consulta. El resultado de la consulta se debe escribir en un archivo.

Por ejemplo:

```
$> ./run-query -DserverAddress=xx.xx.xx.xx:yyyy [ -Dairline=airlineName | -Drunway=runwayName ] -DoutPath=fileName
```

donde

- `run-query` es el script que invoca a la clase del cliente de consulta.
- `xx.xx.xx.xx:yyyy` es la dirección IP y el puerto donde está publicado el servicio de consulta de los despegues.
- Si no se indica `-Dairline` ni `-Drunway` se resuelve la consulta 1.
- Si se indica `-Dairline`, `airlineName` es el nombre de la aerolínea elegida para resolver la consulta 2.
- Si se indica `-Drunway`, `runwayName` es el nombre de la pista elegida para resolver la consulta 3.

- Si se indican ambos `-Dairline` y `-Drunway` la consulta falla y se indica el error de parámetros en pantalla.
- `fileName` es el *path* del archivo de salida con los resultados de la consulta elegida.

De esta forma,

```
$> ./run-query -DserverAddress=10.6.0.1:1099 -DoutPath=query1.csv
```

realiza la consulta de todos los despegues utilizando el servicio remoto publicado en `10.6.0.1:1099` y deja en `query1.csv` los resultados obtenidos según el formato arriba mencionado.

La siguiente invocación

```
$> ./run-query -DserverAddress=10.6.0.1:1099 -DoutPath=../query2.csv
-Dairline='Aerolíneas Argentinas'
```

realiza la consulta de todos los despegues de los vuelos de Aerolíneas Argentinas utilizando el servicio remoto publicado en `10.6.0.1:1099` y deja en `query2.csv` (ubicado en el directorio superior a donde está el intérprete de comandos) los resultados obtenidos según el formato arriba mencionado.

En caso de que no exista ningún despegue que cumpla con el criterio de la consulta, el cliente de consulta deberá imprimir en pantalla un mensaje y no generar el archivo de salida.

Hechos y Consideraciones

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

- Se asume que el formato y contenido del archivo de entrada `vuelos.csv` es correcto y no es necesario validarlo.
- Se puede considerar que el servicio funciona por aeropuerto. Es decir, se prende un servidor para un aeropuerto y al final del día se lo apaga.
- No es necesario que tenga persistencia, debería estar levantado durante toda la operación del aeropuerto sin problemas. Al reiniciar se comienza de cero la misma.

Requisitos

Se requiere implementar:

- Todos los servicios deben ser implementados en **<u>un único *servant*</u>** utilizando RMI y `UnicastRemoteObject`, teniendo en cuenta que los servicios deben poder atender pedidos de clientes de manera concurrente y responder a lo indicado en este enunciado.
- Un servidor que se encargue de instanciar y registrar el *servant* bajo los nombres de los servicios arriba mencionados.
- Los clientes indicados cada uno como una aplicación diferente.

Los servicios deberán informar los errores arrojando las excepciones pertinentes.

<u>Muy Importante:</u>

➔ **Respetar exactamente los nombres de los *scripts*, los nombres de los archivos de entrada y 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**. Tener en cuenta que el archivo de salida debe contener las líneas de encabezado correspondientes indicadas en las salidas de ejemplo.

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.
  - **<u>No se deben entregar los binarios.</u>**
- Un **documento breve** (no más de dos 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: Java 8, Concurrencia y RMI.
- 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: <u>miércoles 01/09</u>**
- **Entrega del trabajo: Estará disponible hasta el <u>jueves 16/09 a las 23:59</u>** la actividad "Entrega TPE 1" localizada en la sección Contenido / Evaluación / TPE 1. En la misma deberán cargar el paquete con el **código fuente** y el **documento**
- **El día <u>miércoles 29/09 a las 18:00</u>** cada grupo tendrá un espacio de 10 minutos para un coloquio. Durante el mismo se les hará una devolución del trabajo, indicando la nota y 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” con la posibilidad de compartir pantalla. Los coloquios se llevarán a cabo en un aula virtual ya creada para cada grupo de Campus. Todos los integrantes del grupo deben acceder primero a la herramienta “Grupos”, elegir su grupo, y luego en “Herramientas del Grupo” elegir la opción “Collaborate”. Por último, deben presionar el botón “Unirse a la sala” para acceder a la sala

del grupo. No será necesario que activen la cámara, sí el micrófono. Todos los integrantes (salvo el primer grupo) deberán estar presentes en la sala del grupo diez minutos antes del horario establecido, ya que el horario es aproximado. El horario de cada grupo será comunicado como anuncio de Campus.

- **El día del recuperatorio será el <u>miércoles 17/11</u>.**
- **<u>No se aceptarán entregas pasado el día y horario establecido como límite.</u>**

Dudas sobre el TPE

Las mismas deben volcarse al Foro de Discusión “TPE” del Campus ITBA.

Recomendaciones

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