Trabajo Práctico Especial 1: Elecciones

03 de Septiembre de 2019

Objetivo

Implementar en **grupos de cuatro personas** un **sistema remoto *thread-safe*** para realizar una elección, contabilizando los votos, definiendo a los ganadores y permitiendo la fiscalización del mismo.

Sistemas de Votación

Se utilizarán tres sistemas de votación distintos:

- **First-Past-The-Post (FPTP)**
  - Breve descripción: El sistema tradicional, donde el candidato que obtuvo más votos es el único ganador.
  - Más información: https://en.wikipedia.org/wiki/First-past-the-post_voting
- **Alternative-Vote (AV)**
  - Breve descripción: Este sistema consiste en un voto rankeado, de forma que el ciudadano puede rankear a los candidatos según su orden de preferencia. En caso de que contabilizando todas las primeras opciones de todos los votos no exista un candidato que alcance la meta de votos necesarios, el candidato que menos votos obtuvo es descartado y esos votos son “reprocesados” y transferidos a otros candidatos teniendo ahora en cuenta la segunda opción. Este ciclo continúa, de ser necesario, para las siguientes opciones del voto hasta que un candidato alcance la meta.
  - Video Recomendado con explicación detallada: https://youtu.be/3Y3jE3B8HsE
  - Más información: https://en.wikipedia.org/wiki/Instant-runoff_voting
- **Single Transferable Vote (STV)**
  - Breve descripción: Al igual que en AV, los ciudadanos emiten un voto rankeado. También los candidatos en el último lugar son eliminados y los porcentajes que tenían son transferidos a otros candidatos en función de las demás opciones que hayan elegido los ciudadanos. La particularidad de STV es que además se desvían los porcentajes de votos de aquellos candidatos que ganaron, es decir, ese porcentaje que sobrepasa el candidato sobre la línea de meta es transferido proporcionalmente a los candidatos de las siguientes opciones.
  - Videos Recomendados con explicaciones detalladas:
    - Video 1: https://youtu.be/l8XOZJkozfI
    - Video 2: https://youtu.be/Ac9070OIMUg
  - Más información: https://en.wikipedia.org/wiki/Single_transferable_vote

72.42 Programación de Objetos Distribuidos

La implementación deberá distinguir tres dimensiones:

- **Nacional:** Donde se elegirá a un partido político ganador para ocupar un cargo ejecutivo nacional, mediante el sistema AV.
- **Provincial:** Donde, en cada provincia, se elegirán cinco partidos políticos ganadores para ocupar los 5 representantes que cada provincia tiene en la legislatura nacional, mediante el sistema STV.
- **Mesa:** Se desea conocer el partido político ganador de cada mesa de votación, sólo para fines estadísticos de consulta, mediante el sistema FPTP.

Para simplificar la implementación se establecen las siguientes concesiones:

- Cada partido político presenta una única lista
- El voto de cada ciudadano cumple las siguiente aserciones:
    - El partido político elegido para la opción abarca ambas dimensiones, nacional y provincial.
    - La primera opción es obligatoria para todos los ciudadanos que emitan un voto, es decir, no existe voto en blanco o cualquier forma de voto anulado/impugnado/etc.
    - Opcionalmente los ciudadanos pueden emitir en el voto rankeado una segunda o hasta una tercera opción, todas diferentes.

Descripción Funcional

Servicios Remotos

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

Servicio de Administración

- Funcionalidad: Apertura y cierre de los comicios
- Usuario: Autoridades gubernamentales
- Debe contar con métodos para:
    - Abrir los comicios. Si la elección ya finalizó, debe arrojar un error.
    - Consultar el estado de los comicios: indicando si están sin iniciar, iniciados o finalizados.
    - Finalizar los comicios. Si las elecciones no se iniciaron, debe arrojar un error.

Servicio de Fiscalización

- Funcionalidad: Registro de los fiscales de los partidos políticos
- Usuario: Fiscales de los partidos políticos
- Debe contar con un método para:
    - Registrar a un fiscal de un partido político en una mesa para que éste sea notificado en el momento en que se registra un voto para su partido político (en cualquiera de las opciones del voto rankeado) en esa mesa en particular. Como sólo se pueden agregar fiscales antes de que inicien los comicios, si la elección está en curso o si

ya finalizó se debe arrojar un error. Para ser notificados cada partido debe proveer un servicio implementando una interfaz de remota provista por este servicio.

Servicio de Votación

- Funcionalidad: Emisión de votos
- Usuario: Ciudadanos
- Debe contar con un método para:
    - Emitir un voto en una mesa de votación. Si los comicios están sin iniciar o si ya finalizaron, debe arrojar un error.

Servicio de Consulta

- Funcionalidad: Consultar los resultados parciales y finales de los comicios
- Usuario: Medios de comunicación
- Debe contar con un método para:
    - Consultar los resultados de la elección. La consulta se puede hacer por cada una de las dimensiones de la elección. La respuesta varía de acuerdo al estado de la elección:
        i. Si los comicios no están abiertos debe arrojar un error.
        ii. Si los comicios están abiertos debe retornar los resultados parciales hasta el momento, siempre utilizando FPTP, sin importar qué dimensión se esté consultando.
        iii. Si los comicios ya finalizaron los resultados serán finales y calculados utilizando el sistema de votación correspondiente para cada dimensión.

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 a la clase del cliente de administración y el resultado se debe imprimir en pantalla.

Por ejemplo:

```
$> java -DserverAddress=xx.xx.xx.xx:yyyy -Daction=actionName
ar.edu.itba.pod.client.ManagementClient
```

donde

- xx.xx.xx.xx:yyyy es la dirección IP y el puerto donde está publicado el servicio de administración de los comicios.

- `actionName` es el nombre de la acción a realizar.
    - `open`: Abre los comicios. Deberá imprimir en pantalla el estado de los comicios luego de invocar a la acción o el error correspondiente.
    - `state`: Consulta el estado de los comicios. Deberá imprimir en pantalla el estado de los comicios al momento de la consulta.
    - `close`: Cierra los comicios. Deberá imprimir en pantalla el estado de los comicios luego de invocar a la acción o el error correspondiente.

De esta forma,

```
$> java -DserverAddress=10.6.0.1:1099 -Daction=open
ar.edu.itba.pod.client.ManagementClient
```

abre los comicios utilizando el servicio remoto publicado en `10.6.0.1:1099` e imprime en pantalla el mensaje correspondiente. Por ejemplo, en caso de que los comicios no estaban abiertos aún se imprime:

```
Election Started
```

Cliente de Fiscalización

La información necesaria para que un fiscal se registre en los comicios se recibe a través de argumentos de línea de comando al llamar a la clase del cliente de fiscalización y el resultado se debe imprimir en pantalla.

Por ejemplo:

```
$> java -DserverAddress=xx.xx.xx.xx:yyyy -Did=pollingPlaceNumber
-Dparty=partyName ar.edu.itba.pod.client.FiscalClient
```

donde

- `xx.xx.xx.xx:yyyy` es la dirección IP y el puerto donde está publicado el servicio de fiscalización.
- `pollingPlaceNumber`: el número de la mesa de votación a fiscalizar.
- `partyName`: el nombre del partido político del fiscal.

De esta forma,

```
$> java -DserverAddress=10.6.0.1:1099 -Did=1001 -Dparty=TIGER
ar.edu.itba.pod.client.FiscalClient
```

registra a un fiscal del partido político `TIGER` en la mesa de votación `1001` utilizando el servicio remoto publicado en `10.6.0.1:1099` e imprime en pantalla un mensaje al momento de la registración y cada vez que se obtiene un voto para el partido político del fiscal (en cualquiera de las opciones del voto rankeado) en la mesa indicada. Por ejemplo, en caso de que los comicios no estaban abiertos aún se imprime:

`Fiscal of TIGER registered on polling place 1001`

y cada vez que se registra un voto para `TIGER` en la mesa `1001` se imprime

`New vote for TIGER on polling place 1001`

Cliente de Votación

Deberá leer de un archivo CSV los votos rankeados de los ciudadanos (indicando el partido político de primera opción, seguido opcionalmente de una segunda y una tercera opción), provenientes de distintas mesas de distintas provincias, donde el delimitador de campos es un punto y coma.

Detalle del archivo de entrada del cliente de votación, a partir de ahora votos.csv

- **Campos:**
  - **El número de mesa** de votación.
  - **El nombre de la provincia de la mesa** de votación.
  - **La/s opción/es que elige el ciudadano en el voto rankeado** por orden de preferencia, separadas por una coma ‘,’.

Cada línea del archivo representa un voto conteniendo los datos de cada uno de los campos mencionados, separados por ;.

Ejemplo de tres líneas de archivo:

```
1000;JUNGLE;LYNX,TIGER
1001;JUNGLE;LYNX
1201;SAVANNAH;SNAKE,TURTLE,BUFFALO
```

- En la primera línea un ciudadano emite un voto en la mesa `1000` de la provincia `JUNGLE` votando como primera opción al partido `LYNX` y como segunda opción al partido `TIGER`.
- En la segunda línea un ciudadano emite un voto en la mesa `1001` de la provincia `JUNGLE` votando como primera y única opción a `LYNX`.
- En la tercera línea un ciudadano emite un voto en la mesa `1201` de la provincia `SAVANNAH` votando como primera opción al partido `SNAKE`, como segunda opción al partido `TURTLE` y como tercera opción al partido `BUFFALO`.

El *path* del archivo `votos.csv`, necesario para emitir una serie de votos, se recibe a través de argumentos de línea de comando al llamar a la clase del cliente de votación y el resultado se debe imprimir en pantalla.

Por ejemplo:

```
$> java -DserverAddress=xx.xx.xx.xx:yyyy -DvotesPath=fileName
ar.edu.itba.pod.client.VoteClient
```

donde

- `xx.xx.xx.xx:yyyy` es la dirección IP y el puerto donde está publicado el servicio de votación.
- `fileName` es el path del archivo de entrada con los votos de los ciudadanos

De esta forma,

```
$> java -DserverAddress=10.6.0.1:1099 -DvotesPath=../votos.csv
ar.edu.itba.pod.client.VoteClient
```

emite los votos presentes en el archivo `votos.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 emitir todos los votos. Por ejemplo, para un archivo `votos.csv` con 100 registros, en caso de que los comicios estaban abiertos se imprime:

```
100 votes registered
```

Cliente de Consulta

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

Consulta 1: Porcentajes por partido político a nivel nacional

Donde cada línea de la salida contenga separados por `;` el porcentaje con dos decimales y el nombre del partido político para la elección a nivel nacional.

El orden de impresión es descendente por porcentaje y luego alfabéticamente por partido político.

En caso de que los comicios aún no abrieron: el archivo no debe crearse porque el cliente de consulta informa el error correspondiente.

- ❏ Salida de ejemplo cuando los comicios ya finalizaron:

```
Porcentaje;Partido
33,00%;OWL
```

Consulta 2: Porcentajes por partido político a nivel provincial

Donde cada línea de la salida contenga separados por `;` el porcentaje con dos decimales y el nombre del partido político para la elección de la provincia indicada.

El orden de impresión es descendente por porcentaje y luego alfabéticamente por partido político.

En caso de que los comicios aún no abrieron el archivo no debe crearse porque el cliente de consulta informa el error correspondiente.

❏ Salida de ejemplo cuando los comicios ya finalizaron:

```
Porcentaje;Partido
20,00%;GORILLA
20,00%;OWL
20,00%;TIGER
20,00%;WHITE_GORILLA
16,00%;TURTLE
```

Consulta 3: Porcentajes por partido político a nivel mesa de votación

Donde cada línea de la salida contenga separados por `;` el porcentaje con dos decimales y el nombre del partido político para la elección en la mesa de votación indicada.

El orden de impresión es descendente por porcentaje y luego alfabéticamente por partido político.

En caso de que los comicios aún no abrieron: el archivo no debe crearse porque el cliente de consulta informa el error correspondiente.

❏ Salida de ejemplo:

```
Porcentaje;Partido
33,00%;OWL
32,00%;GORILLA
16,00%;TIGER
5,00%;TARSIER
4,00%;LYNX
3,00%;BUFFALO
3,00%;WHITE_GORILLA
2,00%;JACKALOPE
1,00%;SNAKE
1,00%;TURTLE
```

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 elección en la dimensión consultada debe escribir en archivo. Además el/los ganador/ganadores de la elección en la dimensión consultada se debe/deben imprimir en pantalla.

Por ejemplo:

```
$> java -DserverAddress=xx.xx.xx.xx:yyyy [ -Dstate=stateName | -Did=pollingPlaceNumber ] -DoutPath=fileName ar.edu.itba.pod.client.QueryClient
```

donde

- `xx.xx.xx.xx:yyyy` es la dirección IP y el puerto donde está publicado el servicio de consulta de los comicios.
- Si no se indica `-Dstate` ni `-Did` se resuelve la consulta 1.

- Si se indica `-Dstate`, `stateName` es el nombre de la provincia elegida para resolver la consulta 2.
- Si se indica `-Did`, `pollingPlaceNumber` es el número de la mesa elegida para resolver la consulta 3.
- Si se indican ambos `-Dstate` y `-Did` 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,

```
$> java -DserverAddress=10.6.0.1:1099 -DoutPath=query1.csv
ar.edu.itba.pod.client.QueryClient
```

realiza la consulta de porcentajes por partido político a nivel nacional utilizando el servicio remoto publicado en `10.6.0.1:1099` y deja en `query1.csv` los resultados obtenidos según el formato arriba mencionado e imprime en pantalla:

```
TIGER won the election
```

La siguiente invocación

```
$> java -DserverAddress=10.6.0.1:1099 -DoutPath=../query2.csv
-Dstate=JUNGLE ar.edu.itba.pod.client.QueryClient
```

realiza la consulta de porcentajes por partido político a nivel provincial 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 e imprime en pantalla:

```
GORILLA, OWL, TIGER, WHITE_GORILLA, TURTLE won the election
```

La implementación debe respetar exactamente el formato de salida enunciado.

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 `votos.csv` es correcto y no es necesario validarlo. En cuanto a las opciones de los votos, no habrán votos con 0 o más de 3 opciones de partidos políticos.
- La cantidad de ganadores a nivel provincial es siempre cinco y no necesita parametrizarse.
- La cantidad de provincias es fija y los posibles valores son: `JUNGLE`, `SAVANNAH`, y `TUNDRA`. Se puede asumir que el archivo de entrada votos.csv menciona únicamente provincias de esta lista y no es necesario validarlo.
- La cantidad de partidos políticos es fija y los posibles valores son: `GORILLA`, `LEOPARD`, `TURTLE`, `OWL`, `TIGER`, `TARSIER`, `MONKEY`, `LYNX`, `WHITE_TIGER`, `WHITE_GORILLA`,

`SNAKE`, `JACKALOPE` y `BUFFALO`. Se puede asumir que el archivo de entrada `votos.csv` menciona únicamente partidos políticos de esta lista y no es necesario validarlo.

- Se puede considerar que el servicio funciona por elección. Es decir, se prende un servidor para una elección y al final se lo apaga.
- No es necesario que tenga persistencia, debería estar levantado durante toda la elección sin problemas.

Requisitos

Se requiere implementar:

- Todos los servicios de la votación y notificación definidos en un único servant 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.

Material a entregar

- 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.
  - No se deben entregar los binarios.
- 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.
- Faltan 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 levanta 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 cliente 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.

Cronograma

- **Presentación del Enunciado: martes 03/09**
- **Hasta el viernes 06/09 inclusive** cada alumno debe inscribirse en un grupo en la sección **Grupos** de Campus ITBA. Si luego de la fecha quedan alumnos sin grupos, éstos serán inscriptos por la cátedra aquellos que aún no estén completos.
- **Antes del miércoles 18/09 a las 23:59** deben cargar el **código fuente** y el **documento** en la actividad "Entrega TPE 1" localizada en la sección Contenido / Evaluación / TPE 1 de Campus ITBA.
- **El día martes 24/09 a las 19:00** cada grupo tendrá 15 minutos para configurar su entorno y mostrar la ejecución de la aplicación a la cátedra. Se tomará nota de las respuestas obtenidas con cada uno de los cientes y esto será parte de la evaluación del trabajo. A criterio de la cátedra también se podrán realizar preguntas sobre la implementación.
- **El día del recuperatorio será el martes 19/11.**
- **No se aceptarán entregas fuera de los días y horarios establecidos.**