Trabajo Práctico Especial 1: Elecciones

02 de Septiembre de 2020

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)**
- **Score then Automatic Runoff (STAR)**
- **Sequential Proportional Approval Voting (SPAV)**

First-Past-The-Post (FPTP)

Breve resumen: El sistema tradicional, donde el candidato que obtiene más votos es el único ganador.

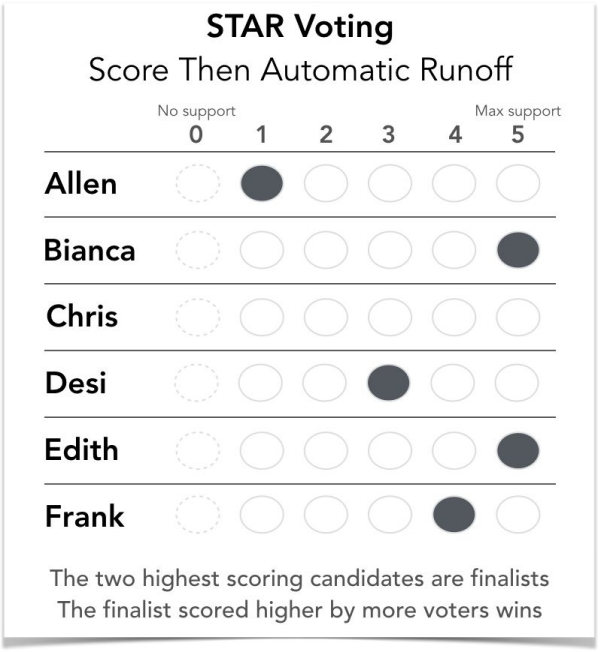
A los efectos del TPE, en caso de un empate entre dos o más candidatos que coincidan en la cantidad de votos, el ganador será el candidato alfabéticamente menor.

Más información: https://en.wikipedia.org/wiki/First-past-the-post_voting

Score then Automatic Runoff (STAR)

Breve resumen: Un sistema de dos etapas para conseguir un único ganador. En la primera, los votantes califican numéricamente a uno o más candidatos. Los puntajes de cada candidato se suman y los dos con mayor puntaje pasan a una segunda ronda "*Runoff*" donde el finalista más preferido en base a todos los votos es el ganador.

El sistema requiere de una boleta donde a cada candidato se le puede asignar un valor numérico entre 0 y 5 inclusive, donde 0 es la peor calificación y 5 la mejor. En caso de no darle una calificación al candidato, se toma como 0. Una boleta puede contar con dos o más candidatos con la misma calificación. Una boleta puede estar vacía y equivale a haber calificado en cero a todos los candidatos de la misma.



Ejemplo de boleta para STAR[1].

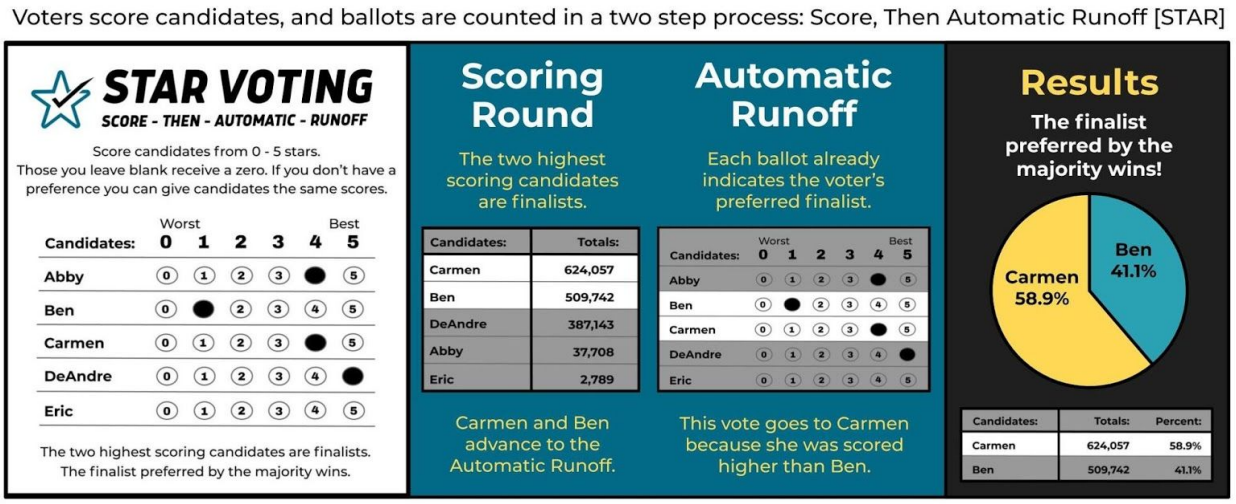
En la primera ronda denominada *scoring round* se suman los puntajes de todos los candidatos de todas las boletas. Supongamos una elección con solo dos boletas, donde en la primera boleta al candidato A se lo califica con un 3 y en la segunda boleta al candidato A se lo califica con un 4, entonces el puntaje total de la primera ronda para el candidato A es 7. Luego de computar los puntajes de todas las boletas, los dos candidatos con mayor puntaje total pasarán a la segunda ronda denominada *automatic runoff*.

A los efectos del TPE, en caso de que se produzca un empate de puntaje total, se selecciona al candidato alfabéticamente menor. Por ejemplo, suponiendo el caso de que el *scoring round* tenga al candidato A con 10 puntos y a los candidatos B y C con 8 puntos y al candidato D con 5 puntos. El primer finalista es el candidato A. Luego, para conseguir el segundo finalista, se desempata por orden alfabético y se elige a B. Así A y B son los finalistas.

En la segunda ronda se computarán todas las boletas que contengan un puntaje mayor estricto a 0 para uno o los dos finalistas. Luego cada una de estas boletas contarán como un voto para el finalista preferido en función del puntaje indicado. Por ejemplo, si una boleta tiene un puntaje de 3 para el finalista A y un puntaje 5 para el finalista B, entonces esa boleta se computará como un voto para el finalista B. A los efectos del TPE, en caso de que una boleta tenga el mismo puntaje positivo para ambos finalistas, el voto se computará para el finalista alfabéticamente menor. Entonces, del conjunto total de boletas, algunas serán descartadas (las que no tengan puntajes positivos para ninguno de los dos finalistas), otras boletas contarán como votos para el primer finalista y otras boletas contarán como votos para el segundo finalista. Luego el finalista con más votos será elegido como ganador. A los efectos del TPE, en caso de empate entre cantidad de votos de ambos finalistas, el ganador será el candidato alfabéticamente menor.

La siguiente infografía resume lo explicado anteriormente:

[1] https://commons.wikimedia.org/wiki/File:STAR_Voting_sample_ballot.png



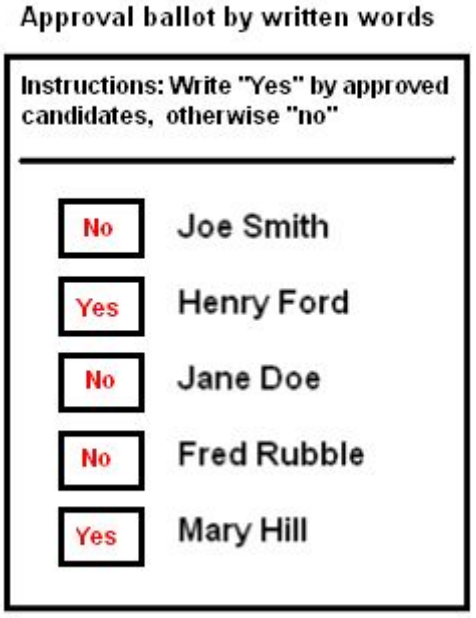
Resumen Funcionamiento STAR[2].

Más información: https://en.wikipedia.org/wiki/STAR_voting y https://www.starvoting.us

Sequential Proportional Approval Voting (SPAV)

Breve resumen: Un sistema de varias rondas donde los ganadores son los candidatos más aprobados.

El sistema requiere de una boleta donde a cada candidato se lo puede aprobar o desaprobar. En caso de no darle una calificación al candidato, se toma como desaprobado. Una boleta puede contar con dos o más candidatos aprobados. Una boleta puede estar vacía y equivale a haber desaprobado a todos los candidatos de la misma.



Ejemplo de boleta para SPAV[3].

[2] https://www.starvoting.us
[3] https://commons.wikimedia.org/wiki/File:Approvalballotword.png

Para elegir N ganadores, el sistema contará con N rondas y en cada ronda, se obtiene un ganador. El funcionamiento consiste en darle un valor a cada una de las boletas. Inicialmente el valor es 1 y luego de cada ronda algunas boletas decrementarán su valor. Por eso el “conteo” en este sistema se basará en proporciones y los resultados finales que acompañarán a cada candidato en lugar de ser enteros estarán dados en valores reales.

En cada ronda se calculará el índice de aprobación de cada candidato como la suma de los índices de aprobación de ese candidato en cada una de las boletas. Si el candidato no es aprobado en una boleta, el índice de aprobación de esa boleta para ese candidato es cero. Si el candidato es aprobado en esa boleta, entonces el índice de aprobación de esa boleta para ese candidato es 1/(1+m) donde m es el número de candidatos aprobados en esa boleta que ya fueron declarados ganadores.

En cada ronda el candidato con mayor índice de aprobación es elegido como ganador de la ronda. Una vez que es elegido como ganador, un candidato no puede volver a ser elegido como ganador en las rondas subsiguientes. El cambio de valor de las boletas se conoce como *reweighting* y está basado en el *D'Hondt method*.

A los efectos del TPE, en caso de que exista un empate en una ronda, es decir hay dos o más candidatos con el mismo índice de aprobación, el ganador de la ronda será el alfabéticamente menor.

Más información: https://en.wikipedia.org/wiki/Sequential_proportional_approval_voting

Implementación

La implementación deberá distinguir tres dimensiones:

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

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 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 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 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 remota provista por este servicio.

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 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
```

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 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,

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

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 Votación

Deberá leer de un archivo CSV los votos de los ciudadanos 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** separadas por una coma ‘,’. Cada opción consiste en el nombre del partido y un valor numérico separados por un *pipe* ‘|’. De esta forma, una línea del CSV modela al mismo tiempo a las tres boletas, una para cada sistema.
        - Para generar una boleta para STAR debe considerar a los valores numéricos como el puntaje de cada candidato.
        - Para generar una boleta para SPAV debe considerar a todos los candidatos mencionados como aprobados.

- Para generar una boleta para FPTP debe considerar únicamente el candidato indicado en el último campo. Se garantiza que este candidato es uno de los candidatos presentes en la boleta (de los considerados para los sistemas anteriores). Esta opción es para fines estadísticos.

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;TIGER|3,LEOPARD|2,LYNX|1;TIGER
1001;JUNGLE;LYNX|1,TIGER|1,LEOPARD|2;LYNX
1002;SAVANNAH;TIGER|3,LYNX|3,OWL|3,BUFFALO|5;BUFFALO
```

- En la primera línea un ciudadano emite un voto en la mesa `1000` de la provincia `JUNGLE`
  - STAR: Puntaje 3 para Tiger, puntaje 2 para Leopard y puntaje 1 para Lynx. Los demás partidos políticos tienen un puntaje 0.
  - SPAV: Tiger, Leopard y Lynx son los candidatos aprobados. Los demás candidatos están desaprobados.
  - FPTP: Voto para Tiger.
- En la segunda línea un ciudadano emite un voto en la mesa `1001` de la provincia `JUNGLE`
  - STAR: Puntaje 2 para Leopard, puntaje 1 para Lynx y puntaje 1 para Tiger. Los demás partidos políticos tienen un puntaje 0.
  - SPAV: Tiger, Leopard y Lynx son los candidatos aprobados. Los demás candidatos están desaprobados.
  - FPTP: Voto para Lynx.
- En la tercer línea un ciudadano emite un voto en la mesa `1002` de la provincia `SAVANNAH`
  - STAR: Puntaje 5 para Buffalo, puntaje 3 para Tiger, puntaje 3 para Lynx y puntaje 3 para Owl. Los demás partidos políticos tienen un puntaje 0.
  - SPAV: Tiger, Lynx, Owl y Buffalo son los candidatos aprobados. Los demás candidatos están desaprobados.
  - FPTP: Voto para 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:

```
$>  ./run-vote -DserverAddress=xx.xx.xx.xx:yyyy -DvotesPath=fileName
```

donde

- `run-vote` es el script que invoca a la clase del cliente de votación.
- `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,

```
$> ./run-vote -DserverAddress=10.6.0.1:1099 -DvotesPath=../votos.csv
```

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 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:

```
$> ./run-fiscal -DserverAddress=xx.xx.xx.xx:yyyy -Did=pollingPlaceNumber
-Dparty=partyName
```

donde

- `run-fiscal` es el script que invoca a la clase del cliente de fiscalización.
- `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,

```
$>  ./run-fiscal -DserverAddress=10.6.0.1:1099 -Did=1001 -Dparty=TIGER
```

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. Cada vez que se obtiene un voto en la mesa indicada donde el partido político del fiscal coincide con el partido de la opción de FPTP del voto se emite un mensaje.

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 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

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

En caso de que los comicios estén en curso, cada línea de la salida debe contener separados por **;** el porcentaje con dos decimales y el nombre del partido político para la elección a nivel nacional. Esta consulta es para fines estadísticos y consistirá en el relevamiento de todas las mesas de votación (usando FPTP).

En caso de que los comicios ya finalizaron, se deben listar los resultados de las dos rondas: *scoring round* y *automatic runoff*. Por último se listará el nombre del partido ganador.

- *Scoring Round:* cada línea de la salida debe contener separados por **;** el puntaje entero y el nombre del partido político. El orden de impresión es descendente por puntaje y luego alfabéticamente por partido político.
- *Automatic Runoff:* cada línea de la salida debe contener separados por **;** el porcentaje con dos decimales y el nombre del partido político. El orden de impresión es descendente por porcentaje y luego alfabéticamente por partido político.

❏ Salida de ejemplo con los comicios en curso:

```
Percentage;Party
66.67%;TURTLE
33.33%;LYNX
```

❏ Salida de ejemplo cuando los comicios ya finalizaron:

```
Score;Party
18;LYNX
18;TURTLE
12;LEOPARD
4;JACKALOPE
Percentage;Party
66.67%;LYNX
33.33%;TURTLE
Winner
LYNX
```

Consulta 2: Porcentajes por partido político a nivel provincial

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

En caso de que los comicios estén en curso, cada línea de la salida debe contener separados por `;` el porcentaje con dos decimales y el nombre del partido político para la elección a nivel provincial. Esta consulta es para fines estadísticos y consistirá en el relevamiento de todas las mesas de votación de la provincia indicada (usando FPTP).

En caso de que los comicios ya finalizaron, se deben listar los resultados de cada ronda. Para una ronda, cada línea de la salida debe contener separados por "`;`" el índice de aprobación con dos decimales y el nombre del partido político. Por último se listará el nombre de los partidos ganadores al finalizar esa ronda, separador por ´,´.

- Salida de ejemplo con los comicios en curso:

```
Percentage;Party
33.33%;LEOPARD
33.33%;TIGER
33.33%;TURTLE
```

- Salida de ejemplo cuando los comicios ya finalizaron:

```
Round 1
Approval;Party
2.00;LEOPARD
2.00;TURTLE
1.00;LYNX
1.00;TIGER
Winners
LEOPARD
Round 2
Approval;Party
1.50;TURTLE
1.00;LYNX
0.50;TIGER
Winners
LEOPARD,TURTLE
Round 3
Approval;Party
0.50;LYNX
0.33;TIGER
Winners
LEOPARD,TURTLE,LYNX
```

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

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

En caso de que los comicios estén en curso o ya finalizaron, cada línea de la salida debe contener 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. Por último se listará el nombre del partido ganador.

- Salida de ejemplo:

```
Percentage;Party
45.90%;LYNX
39.34%;LEOPARD
14.75%;TIGER
Winner
LYNX
```

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 se debe escribir en un archivo.

Por ejemplo:

```
$> ./run-query -DserverAddress=xx.xx.xx.xx:yyyy [ -Dstate=stateName | -Did=pollingPlaceNumber ] -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 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,

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

realiza la consulta 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.

La siguiente invocación

```
$> ./run-query -DserverAddress=10.6.0.1:1099 -DoutPath=../query2.csv -Dstate=JUNGLE
```

realiza la consulta a nivel provincial para la provincia `JUNGLE` 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.

La implementación debe respetar exactamente el formato de salida enunciado. Contemplar que:

- Los números y porcentajes reales deben ser redondeados a dos cifras decimales y siempre se deben mostrar los dos decimales.
- Sólo se deben listar los partidos que hayan tenido algún voto en los comicios.

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. Se garantiza que en la sección de opciones de voto de cada línea existirá al menos un partido político acompañado de un entero mayor a cero. Es decir, `1001;JUNGLE;LYNX|1;LYNX` lo que permite generar un voto válido para cada sistema de votación. También se garantiza que no existirán opciones con 0 (es decir, `...;LYNX|0;...`)
- La cantidad de ganadores a nivel provincial es siempre tres 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: `TIGER`, `LEOPARD`, `LYNX`, `TURTLE`, `OWL`, `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 luego de consultados los resultados, se lo apaga sin persistir la información.
- No es necesario que tenga persistencia, debería estar levantado durante toda la elección sin problemas.
- En caso de que se realice una consulta sobre una dimensión con cero votos, la aplicación debe lanzar una excepción con el mensaje **"**`No Votes`**"**.
- Como la cantidad de partidos políticos es fija no es necesario validar que la cantidad de ganadores pedida pueda resultar mayor a la cantidad de partidos políticos.

Requisitos

Se requiere implementar:

- Todos los servicios (votación y notificación) deben ser implementados 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.

Muy Importante:

> ➔ **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 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.
  - 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

Revisar que en el archivo comprimido también se encuentra el directorio oculto .git/ donde se detallan todas las modificaciones realizadas.

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 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 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.

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 si se lo solicitase.**

Cronograma

- **Presentación del Enunciado: miércoles 02/09**
- **Antes del jueves 17/09 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 miércoles 23/09 a las 19:00** cada grupo tendrá un espacio de 15 minutos para mostrar la ejecución de la aplicación a la cátedra. Para ello un integrante del equipo deberá compartir pantalla. Se tomará nota de las respuestas obtenidas con los clientes y esto será parte de la evaluación del trabajo. A criterio de la cátedra también se podrán realizar preguntas sobre la implementación.
  Los coloquios se llevarán a cabo en un aula virtual ya creada para cada grupo de Campus. Durante el mismo se les hará una devolución del trabajo, indicando la nota y los principales errores cometidos. 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 miércoles 18/11.**
- **No se aceptarán entregas pasado el día y horario establecido como límite.**

Dudas sobre el TPE

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

Recomendaciones

72.42 Programación de Objetos Distribuidos

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