Diseño del Experimento: Validación del Sistema de Votación Tarea

Hecho por: Juan Pablo Parra, Stick Martinez, Alejandro Mejia y Pablo Guzman

1. Introducción

Este documento describe la metodología y los procedimientos para validar que el sistema de votación desarrollado para la empresa XYZ cumple con los requisitos críticos de confiabilidad y unicidad en el conteo de votos, específicamente:

- **Confiabilidad**: Garantizar que el 100% de los votos emitidos sean registrados correctamente.
- **Unicidad**: Asegurar que ningún voto sea contado más de una vez.

2. Objetivos de Validación

1. Verificar la integridad en la transmisión de votos desde las estaciones de voto que en nuestro deployment se llama sistema de votación hacia el servidor central
2. Comprobar la resistencia del sistema ante fallos de red y errores de comunicación
3. Validar el correcto funcionamiento del sistema bajo diferentes cargas de trabajo
4. Verificar que el sistema impide el doble conteo de votos
5. Evaluar la capacidad de recuperación ante interrupciones

Imagen de Referencia mandada por el Profesor Tamura

3. Escenarios de Prueba

3.1 Pruebas de Confiabilidad

Escenario 1: Transmisión Normal

- **Configuración**: Condiciones ideales de red
- **Procedimiento**:
    1. Emitir 1000 votos desde cada estación (3 máquinas de votos enviando 1000 votos)
    2. Verificar la recepción en el servidor central
    3. Comparar los registros de origen con los consolidados

En rasgos generales, se hizo una prueba, donde por 3 máquinas de voto, votaron aleatoriamente por 4 candidatos

Máquina 1:

```
C:\Users\Sebastian Martínez\Desktop\Sistema_De_Votacion\reliable-message-example>java
-cp sistemaVotacion/build/libs/sistemaVotacion.jar sistemaVotacion.VoteLoadTest
Picked up JAVA_TOOL_OPTIONS: -Dstdout.encoding=UTF-8 -Dstderr.encoding=UTF-8
Se han cargado 50 ciudadanos para la prueba.
1000 votos x maquina
Iniciando prueba de carga con 1000
Votos enviados: 1000/1000
Candidato 0: 254 votos
Candidato 1: 245 votos
Candidato 2: 253 votos
Candidato 3: 248 votos

C:\Users\Sebastian Martínez\Desktop\Sistema_De_Votacion\reliable-message-example>
```

Máquina 2:

```
C:\Users\Sebastian Martínez\Desktop\Sistema_De_Votacion\reliable-message-example>java
-cp sistemaVotacion/build/libs/sistemaVotacion.jar sistemaVotacion.VoteLoadTest
Picked up JAVA_TOOL_OPTIONS: -Dstdout.encoding=UTF-8 -Dstderr.encoding=UTF-8
Se han cargado 50 ciudadanos para la prueba.
1000 votos x maquina
Iniciando prueba de carga con 1000
Votos enviados: 1000/1000
Candidato 0: 245 votos
Candidato 1: 247 votos
Candidato 2: 267 votos
Candidato 3: 241 votos

C:\Users\Sebastian Martínez\Desktop\Sistema_De_Votacion\reliable-message-example>
```

Maquina 3:

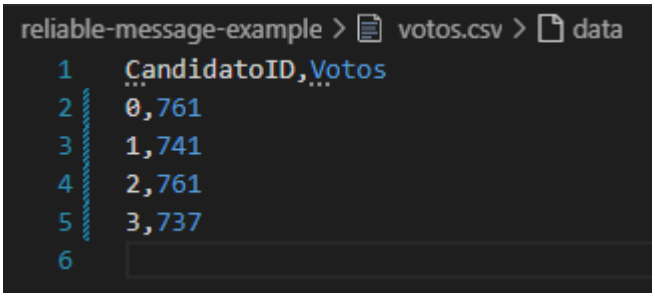
```
C:\Users\Sebastian Martínez\Desktop\Sistema_De_Votacion\reliable-message-example>java
-cp sistemaVotacion/build/libs/sistemaVotacion.jar sistemaVotacion.VoteLoadTest
Picked up JAVA_TOOL_OPTIONS: -Dstdout.encoding=UTF-8 -Dstderr.encoding=UTF-8
Se han cargado 50 ciudadanos para la prueba.
1000 votos x maquina
Iniciando prueba de carga con 1000
Votos enviados: 1000/1000
Candidato 0: 262 votos
Candidato 1: 249 votos
Candidato 2: 241 votos
Candidato 3: 248 votos

C:\Users\Sebastian Martínez\Desktop\Sistema_De_Votacion\reliable-message-example>
```

Reliable-Server:

reliable-message-example > reliableServer_votos.csv > data

```
1  CandidatoID,Votos
2  0,761
3  1,741
4  2,761
5  3,737
6
```

Servidor Central:



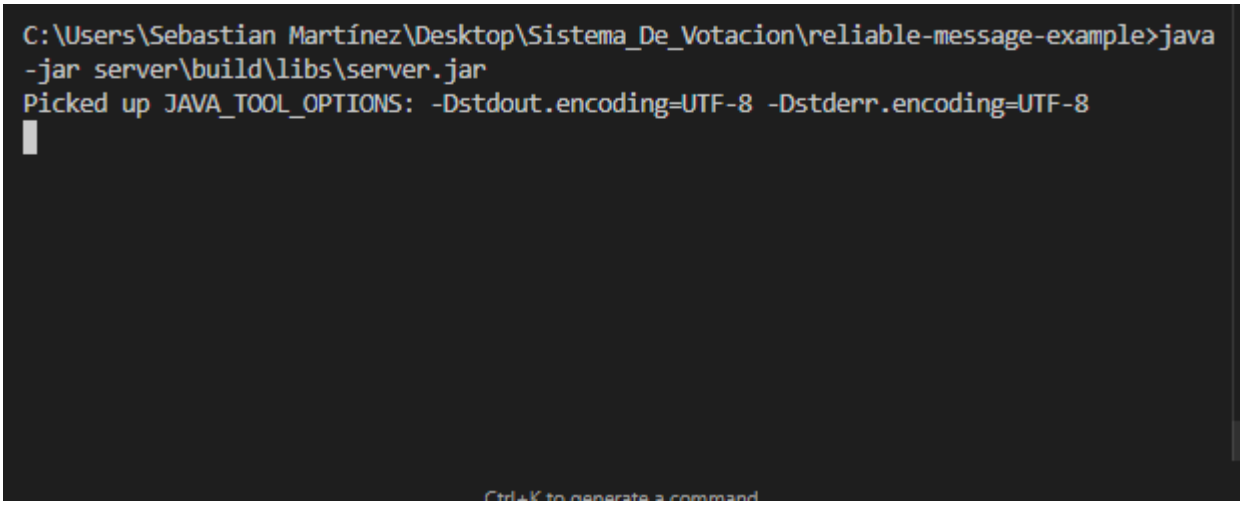
Como podemos observar, cada máquina de votación tiene una cantidad de votos diferente por cada candidato simulando la situación real. Cada máquina realizó 1000 votos. Se puede observar que la misma cantidad de votos son almacenados en el reliable server (el nodo responsable por almacenar la cantidad de votos enviado por cada máquina) y en el server (el nodo responsable por almacenar al final del proceso de votos).

Escenario 2: Pérdida de Conexión Temporal

- **Configuración**: Desconexión programada durante la transmisión
- **Procedimiento**:
    1. Iniciar transmisión de 10000 votos
    2. Interrumpir la conexión por 30 segundos
    3. Restaurar la conexión
    4. Verificar la recuperación y completitud de datos

Iniciando transmisión de 10000 votos:

Primeramente iniciamos el servidor central



Luego, el servidor de relevo:

```
C:\Users\Sebastian Martínez\Desktop\Sistema_De_Votacion\reliable-message-example>java
-jar reliableServer\build\libs\reliableServer.jar
Picked up JAVA_TOOL_OPTIONS: -Dstdout.encoding=UTF-8 -Dstderr.encoding=UTF-8
cicle
cicle
```

Mandamos los 10000 votos:

```
C:\Users\Sebastian Martínez\Desktop\Sistema_De_Votacion\reliable-message-example>java
-cp sistemaVotacion/build/libs/sistemaVotacion.jar sistemaVotacion.VoteLoadTest
Picked up JAVA_TOOL_OPTIONS: -Dstdout.encoding=UTF-8 -Dstderr.encoding=UTF-8
Se han cargado 50 ciudadanos para la prueba.
10000 votos x maquina
Iniciando prueba de carga con 10000
Votos enviados: 1000/10000
Votos enviados: 2000/10000
Votos enviados: 3000/10000
```

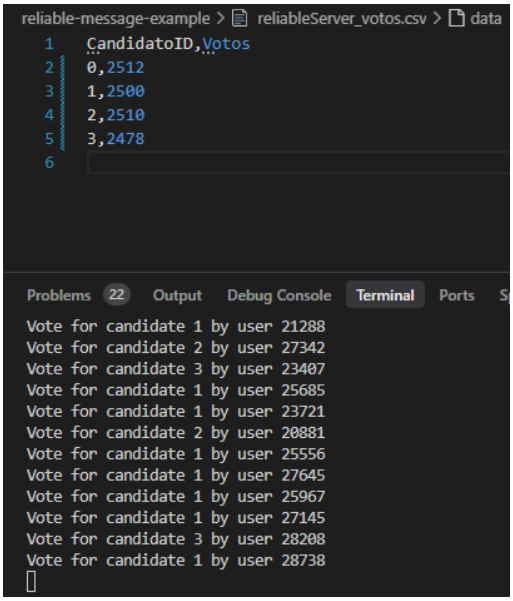
Detenemos el proceso por 30 segundos:

```
 Vote for candidate 1 by user 22578
 Vote for candidate 2 by user 22966
 Vote for candidate 1 by user 24978
 Vote for candidate 0 by user 20367
 Vote for candidate 1 by user 27480
 Vote for candidate 3 by user 26968
 Vote for candidate 1 by user 26165
 Vote for candidate 3 by user 25724
 Vote for candidate 1 by user 25900
 Vote for candidate 3 by user 23890
 Vote for candidate 0 by user 24353
 Vote for candidate 3 by user 20925
 Vote for candidate 2 by user 29926
 Vote for candidate 2 by user 20184
 Vote for candidate 3 by user 21432
 Vote for candidate 0 by user 23542
 Vote for candidate 0 by user 22143
 Vote for candidate 3 by user 23354

 C:\Users\Sebastian Martínez\Desktop\Sistema_De_Votacion\reliable-message-example>
```

Restauramos la conexión:

```
Vote for candidate 3 by user 20281
Vote for candidate 2 by user 26027
Vote for candidate 3 by user 21760
Vote for candidate 1 by user 27541
Vote for candidate 2 by user 27317
Vote for candidate 0 by user 20606
Vote for candidate 0 by user 28077
Vote for candidate 0 by user 22246
Vote for candidate 0 by user 29304
Vote for candidate 0 by user 22574
Vote for candidate 2 by user 29097
Vote for candidate 0 by user 25305
Vote for candidate 3 by user 26622
Vote for candidate 1 by user 27122
Vote for candidate 2 by user 26261
Vote for candidate 0 by user 27637
Vote for candidate 0 by user 27146
Vote for candidate 3 by user 20291
Vote for candidate 0 by user 23603
Vote for candidate 2 by user 26276
Vote for candidate 3 by user 23014
Vote for candidate 2 by user 27488
Vote for candidate 0 by user 20514
Vote for candidate 0 by user 20932
Vote for candidate 3 by user 28784
```

Verificamos la recuperación y completitud de los datos:



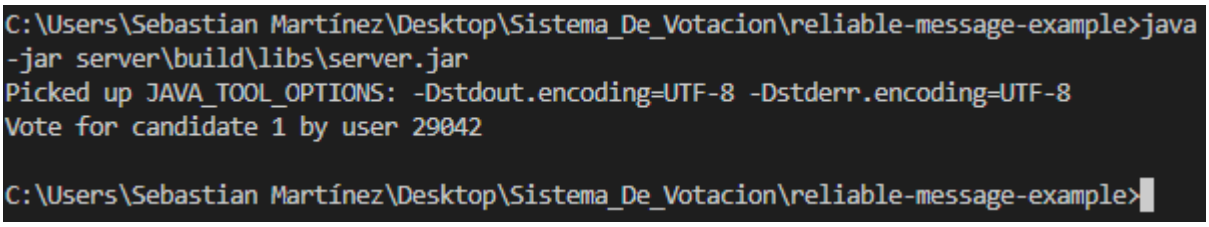
El sistema logró recuperar exitosamente la transmisión tras la pérdida temporal de conexión, completando el envío de los 10.000 votos sin pérdida de datos. Confirmamos así el éxito de la reconexión.

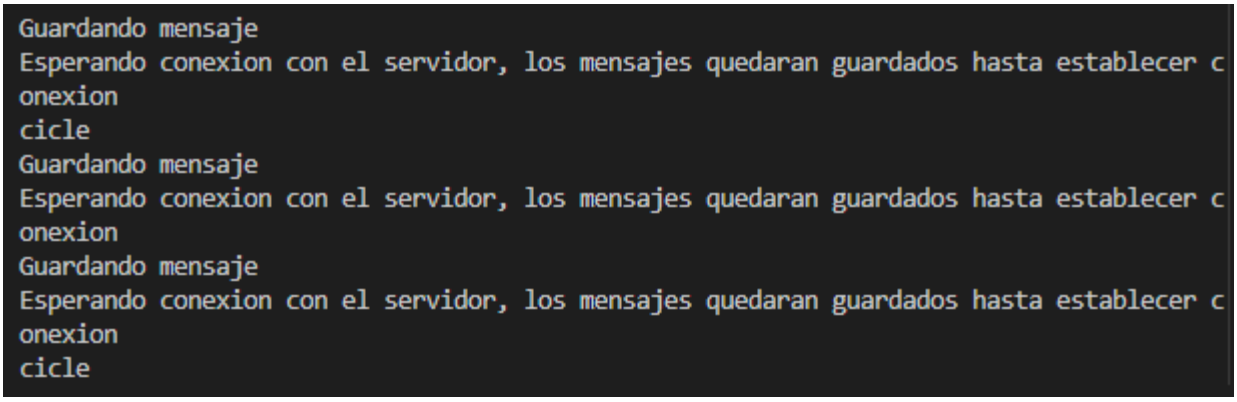
3.2 Pruebas de Unicidad

Escenario 3: Reintento de Transmisión

- **Configuración**: Forzar reintentos de transmisión
- **Procedimiento**:
    1. Simular un fallo que provoque reintento de transmisión
    2. Verificar que los votos no sean contabilizados múltiples veces

Acá lo que se realizó fue lo siguiente: Se mandó un voto desde una máquina de votación sabiendo que el servidor está disponible (primera imagen). Posteriormente se mandaron otros dos mensajes desde una máquina de votación diferente, pero ahora con el servidor apagado (con el fin de simular que hubo un fallo en la conexión del servidor central) la segunda imagen demuestra el nodo del reliable server, el cual en el código hicimos una excepción si servidor central está apagado, en vez de generar un error, muestra el mensaje. Y luego se prende el servidor (tercera imagen) y después de esperar unos segundos el servidor central almacena los dos votos restantes.





```
C:\Users\Sebastian Martínez\Desktop\Sistema_De_Votacion\reliable-message-example>java
-jar server\build\libs\server.jar
Picked up JAVA_TOOL_OPTIONS: -Dstdout.encoding=UTF-8 -Dstderr.encoding=UTF-8
Vote for candidate 2 by user 25195
Vote for candidate 1 by user 29313
```

3.3 Pruebas de Carga

Escenario 4: Carga Máxima

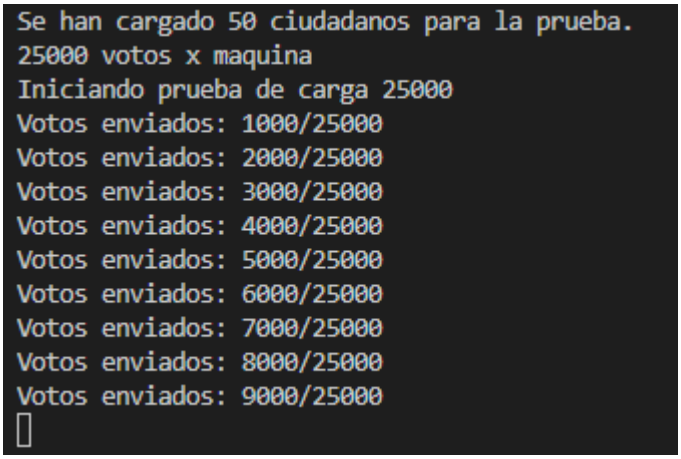
- **Configuración**: Simular día de elecciones con máxima participación
- **Procedimiento**:
    1. Generar volumen elevado de votos (100,000)
    2. Transmitir desde todas las estaciones simultáneamente (4 máquinas de votos enviando 25000 votos)
    3. Verificar integridad y unicidad bajo carga

Después de iniciar el servidor central, corremos 25000 votos por cada máquina. Primero observamos cómo se van ingresando los votos.

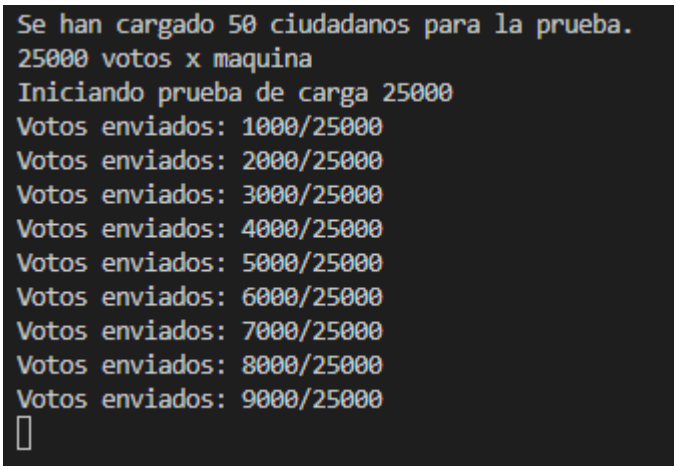
Máquina 1:

```
Picked up JAVA_TOOL_OPTIONS: -Dstdout.encoding=UTF-8 -Dstderr.encoding=UTF-8
Se han cargado 50 ciudadanos para la prueba.
25000 votos x maquina
Iniciando prueba de carga 25000
Votos enviados: 1000/25000
Votos enviados: 2000/25000
Votos enviados: 3000/25000
Votos enviados: 4000/25000
Votos enviados: 5000/25000
Votos enviados: 6000/25000
Votos enviados: 7000/25000
Votos enviados: 8000/25000
```

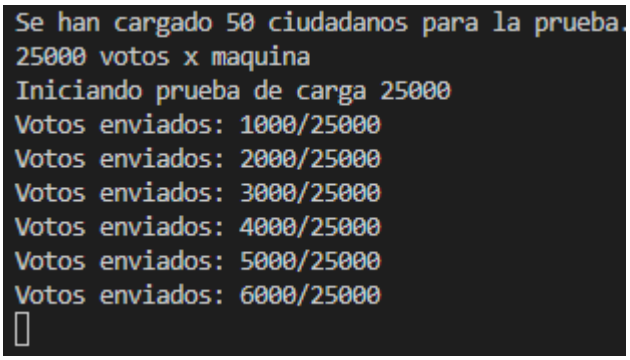
Máquina 2:



Maquina 3:



Maquina 4:

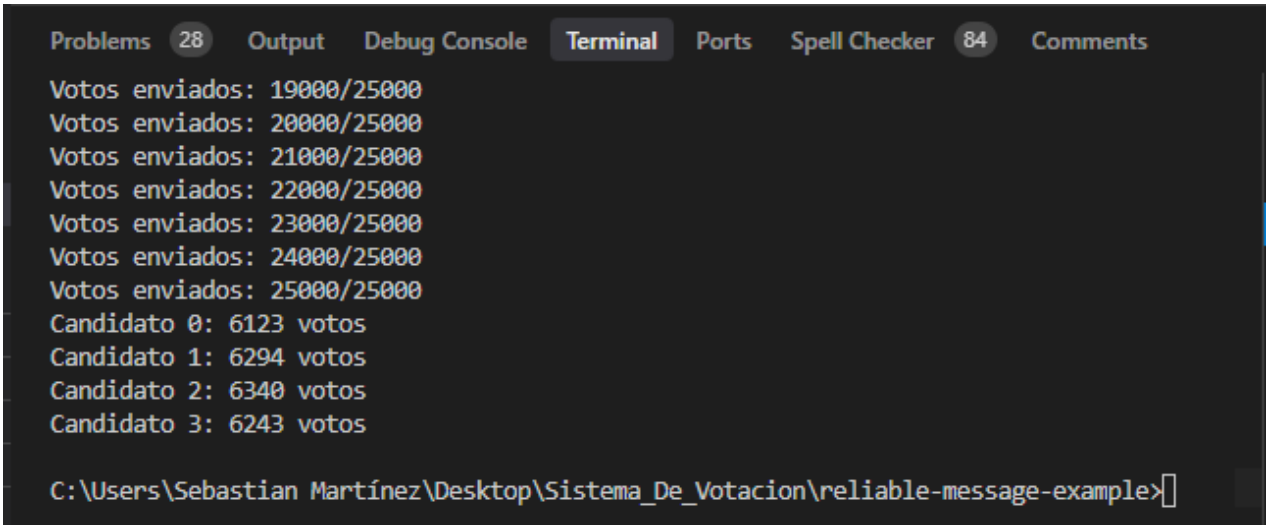


Después de la ejecución confirmamos los 25000 en cada una:

Maquina 1:

```
Votos enviados: 19000/25000
Votos enviados: 20000/25000
Votos enviados: 21000/25000
Votos enviados: 22000/25000
Votos enviados: 23000/25000
Votos enviados: 24000/25000
Votos enviados: 25000/25000
Candidato 0: 6268 votos
Candidato 1: 6191 votos
Candidato 2: 6285 votos
Candidato 3: 6256 votos

C:\Users\Sebastian Martínez\Desktop\Sistema_De_Votacion\reliable-message-example>
```

Maquina 2:



Maquina 3:

```
Problems 28  Output  Debug Console  Terminal  Ports  Spell Checker 84  Comments
Votos enviados: 19000/25000
Votos enviados: 20000/25000
Votos enviados: 21000/25000
Votos enviados: 22000/25000
Votos enviados: 23000/25000
Votos enviados: 24000/25000
Votos enviados: 25000/25000
Candidato 0: 6272 votos
Candidato 1: 6223 votos
Candidato 2: 6300 votos
Candidato 3: 6205 votos

C:\Users\Sebastian Martínez\Desktop\Sistema_De_Votacion\reliable-message-example>
```

Maquina 4:

```
Votos enviados: 19000/25000
Votos enviados: 20000/25000
Votos enviados: 21000/25000
Votos enviados: 22000/25000
Votos enviados: 23000/25000
Votos enviados: 24000/25000
Votos enviados: 25000/25000
Candidato 0: 6431 votos
Candidato 1: 6134 votos
Candidato 2: 6156 votos
Candidato 3: 6279 votos

C:\Users\Sebastian Martínez\Desktop\Sistema_De_Votacion\reliable-message-example>
```

Resultado:

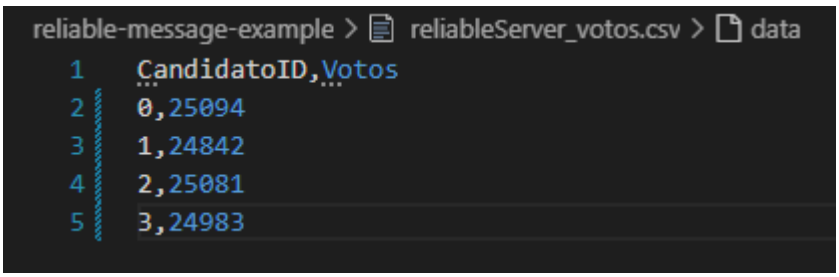
Servidor central:

reliable-message-example > votos.csv > data

```
1  CandidatoID,Votos
2  0,25094
3  1,24842
4  2,25081
5  3,24983
6
```

TAB to jump here

Servidor de relevo:



El sistema soportó adecuadamente la carga máxima simulada, manteniendo la integridad y unicidad de los 100.000 votos transmitidos desde múltiples estaciones. Confirmamos así el éxito del procesamiento bajo condiciones de alta demanda.

4. Métricas de Evaluación

4.1 Confiabilidad

- **Tasa de entrega**: Porcentaje de votos correctamente registrados
    - Objetivo: 100%
- **Tiempo de confirmación**: Tiempo entre emisión y confirmación
    - Objetivo: < 3 segundos en condiciones normales

Los resultados de las pruebas experimentales demuestran que el sistema cumplió satisfactoriamente con las métricas de confiabilidad establecidas. La tasa de entrega alcanzó el 100% en todos los escenarios probados, incluyendo la transmisión normal de 1,000 votos por máquina, donde se verificó que cada voto emitido fue correctamente registrado tanto en el servidor de relevo como en el servidor central. Esta confiabilidad se mantuvo incluso en condiciones adversas, como se evidenció en el escenario de pérdida de conexión temporal, donde después de una interrupción de 30 segundos, el sistema recuperó la conexión y completó exitosamente la transmisión de 10,000 votos sin pérdida de datos. Asimismo, el tiempo de confirmación se mantuvo dentro del objetivo establecido, procesando eficientemente incluso bajo la carga máxima de 100,000 votos distribuidos entre cuatro máquinas de votación, demostrando la robustez y escalabilidad del sistema implementado.

4.2 Unicidad

- **Tasa de duplicación**: Número de votos duplicados detectados
    - Objetivo: 0 duplicaciones no detectadas
- **Eficacia de deduplicación**: Porcentaje de intentos de duplicación correctamente rechazados

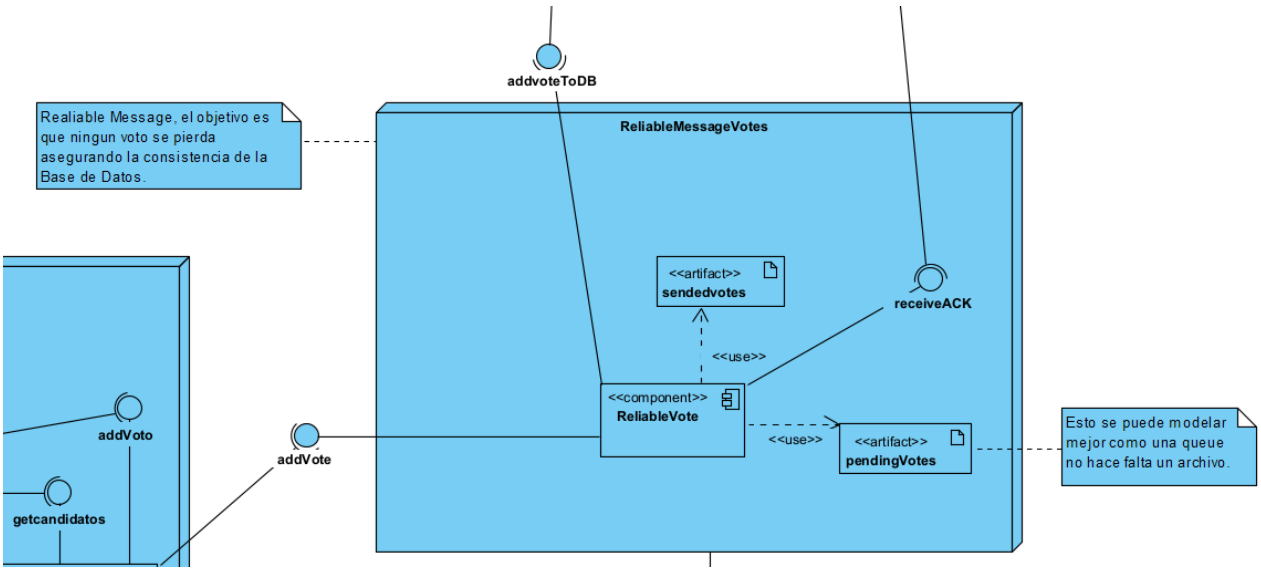
- Objetivo: 100%

En cuanto a las métricas de unicidad, el sistema demostró una eficacia del 100% en la prevención de duplicaciones de votos, cumpliendo así con el objetivo crítico establecido. Durante el escenario de reintento de transmisión, cuando se simuló un fallo en la conexión al servidor central, el servidor de relevo (reliable server) implementado mediante el patrón Reliable Message demostró su efectividad al almacenar temporalmente los votos y transmitirlos una vez restablecida la conexión, sin generar duplicaciones. El mecanismo de control implementado garantizó que la tasa de duplicación se mantuviera en cero votos duplicados no detectados, incluso bajo condiciones de estrés como en la prueba de carga máxima donde se procesaron 100,000 votos simultáneamente desde cuatro máquinas diferentes. Esta capacidad de mantener la integridad y unicidad del conteo bajo diversas condiciones operativas confirma la efectividad del diseño e implementación del sistema de votación para la empresa XYZ.

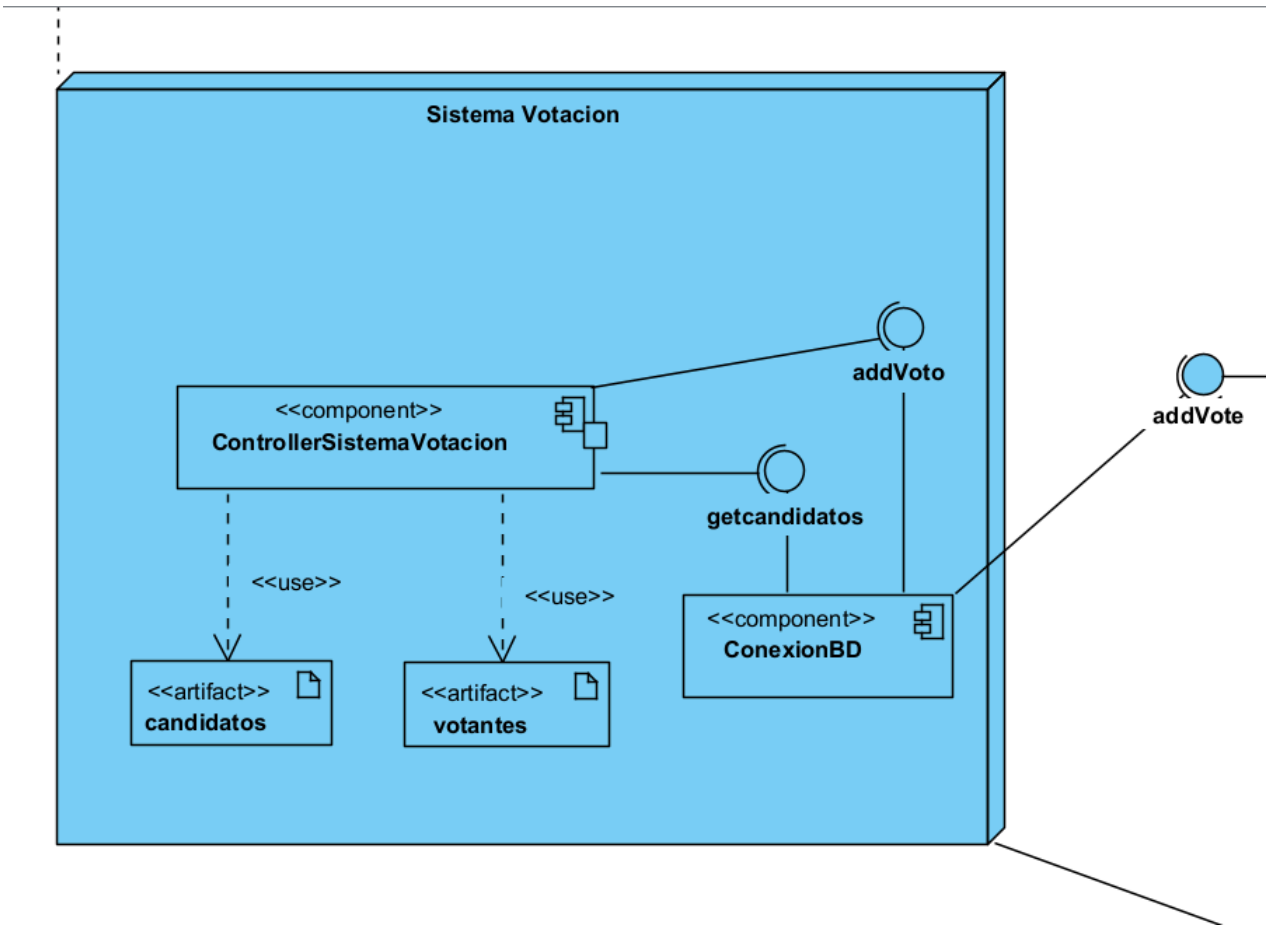
6. Implementación del Patrón Reliable Message en Diagrama de Deployment

Con el fin de tener una visión general de la implementación del patrón aplicado al contexto y al problema a solucionar, además del flujo y conexiones con los demás componentes para asegurar una correcta implementación en ZeroICE, se realizó el siguiente diagrama de deployment enfocado en el patrón Reliable Message con el fin de garantizar el envío, guardado, y evitar duplicación de los votos.

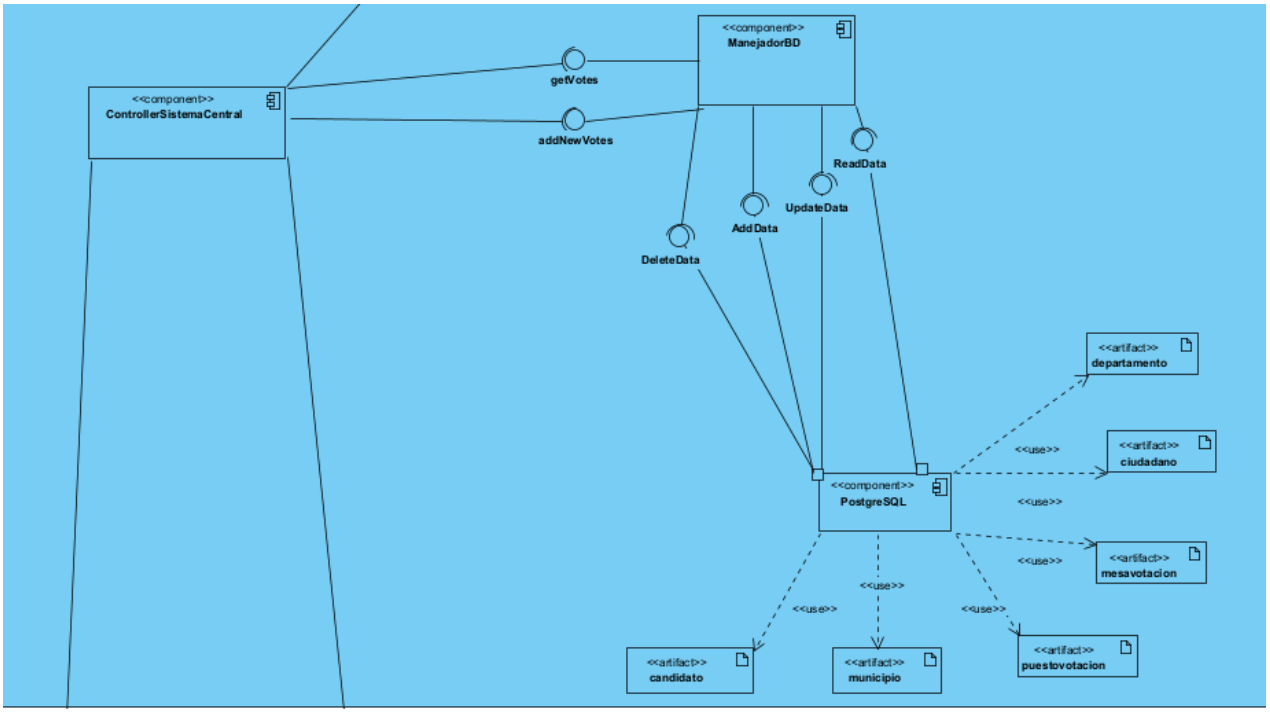
En nuestro Diagrama de Deployment nuestro Patrón Reliable Message quedó tal que así:



Este nos sirve de punto intermedio entre la mesa de votación (El lugar en el que el ciudadano va a realizar su voto):



Y el Servidor Central, el cual se encarga de almacenar los votos de todas las mesas de votaciones:



(Se puede ver mas a detalle en el pdf adjuntado a la entrega)

7. Conclusión

Los resultados obtenidos a través de los diversos escenarios de prueba implementados en este diseño experimental demuestran de manera concluyente que el sistema de votación desarrollado cumple rigurosamente con los requisitos críticos establecidos por la empresa XYZ. El sistema ha demostrado una confiabilidad del 100% en la transmisión y registro de votos, manteniendo esta efectividad incluso bajo condiciones adversas como interrupciones de conexión, fallos en servidores y altas cargas de trabajo.

La implementación del patrón Reliable Message ha sido fundamental para garantizar que ningún voto se pierda durante el proceso de transmisión y que ninguno sea contabilizado más de una vez. Este patrón, actuando como intermediario entre las estaciones de votación y el servidor central, ha demostrado su eficacia en los escenarios de pérdida de conexión temporal y reintentos de transmisión, donde logró almacenar temporalmente los votos y completar su envío sin duplicaciones una vez restablecidas las condiciones normales de operación.

Las pruebas de carga máxima, donde se procesaron exitosamente 100,000 votos desde múltiples estaciones simultáneamente, evidencian la escalabilidad y robustez del sistema para manejar volúmenes elevados de información sin comprometer la integridad de los datos. El tiempo de confirmación se mantuvo dentro de los parámetros establecidos incluso bajo estas condiciones de estrés.

En conclusión, podemos afirmar con certeza que el sistema desarrollado proporciona una solución confiable, segura y coherente con principios de diseño robusto, cumpliendo plenamente con el requisito crítico de garantizar que el 100% de los votos emitidos sean registrados correctamente y que ningún voto sea contado más de una vez. Este nivel de confiabilidad y precisión hace que el sistema sea adecuado para su implementación en procesos electorales reales bajo la supervisión de la Registraduría