David Artunduaga, Rony Ordoñez, Juan de la Pava, Diego Polanco

Diseño de experimento

A continuación, se describe de manera detallada el Diseño del Experimento para validar que, dentro de la arquitectura Nodo de votación → Lugar de votación → Servicio de votación → Base de datos, se cumplan estrictamente los requisitos de confiabilidad (100% de votos registrados) y unicidad (ningún voto contado más de una vez). Este diseño se basa en la aclaración de que la comunicación interna dentro del Lugar de votación y los nodos individuales es confiable, pero la comunicación externa (Lugar de votación al Servicio de votación) puede no serlo.

Objetivos del experimento

Validar Confiabilidad (Tolerancia a Fallos en la Conexión Externa)

- Garantizar que, incluso si la conexión entre el "Lugar de votación" y el "Servicio de votación" falla intermitentemente, todos los votos generados en los Nodos de votación terminen persistidos en la Base de datos del Servicio.

Verificar Unicidad (Idempotencia y Detección de Duplicados)

- Asegurar que, sin importar la cantidad de reenvíos (por reintentos o duplicaciones accidentales), el "Servicio de votación" persista cada voto exactamente una sola vez en la tabla de votos.

Confirmar Atributos de Calidad Específicos

- Consistencia: los resultados en la Base de datos (conteos, CSV finales) reflejen fielmente los votos únicos sin pérdidas ni duplicaciones.

Explicación de la arquitectura

Nodo de votación (CLI local)

- Ubicado en cada máquina de votación (por ejemplo, un terminal de la sala de cómputo).
- Provee una interfaz de línea de comandos para que el operador o ciudadano ingrese su número de documento y seleccione el candidato.
- Construye internamente un objeto de voto que incluye:
    - Documento de identidad (String).
    - Identificador del candidato (entero).
    - Marca de tiempo local (timestamp).

Se comunica de forma directa con el "Lugar de votación" para iniciar la validación (para la entrega final) y envío del voto.

La comunicación entre el Nodo y el Lugar se considera 100 % confiable.

Lugar de votación (Servidor ICE local)

Cada estación (mesa) ejecuta un proceso que expone la interfaz ICE VotingSite con el método sendVote(Vote vote).

Funciones principales:

- Validación local del documento (por esta entrega no).
- Control de doble votación local:
    - Mantiene un conjunto en memoria (cedulasVotadasLocal) con los documentos que ya votaron en esa estación.
    - Si el documento ya figura en ese conjunto, rechaza sin llamar al servicio central.
- Lógica de envío confiable (Reliable Messaging) hacia el Servicio de votación:
    - Construye un mensaje que encapsula el objeto de voto (incluyendo mesaId y candidatoId).
    - Intenta invocar al Servicio central (ICE) para persistir el voto.
    - Si no recibe respuesta (timeout o excepción), reintenta (relyable message).

Servicio de votación (Servidor central)

Expone la interfaz ICE VotingService

Recepción de cada voto:

- Si no existe, lo inserta en la tabla Votos y retorna true a la estación.
- Si ya existe (voto duplicado), retorna false y no lo inserta nuevamente.

Persistencia en Base de datos:

Usa un esquema relacional (por ejemplo, PostgreSQL) con una tabla Votos que incluye columnas obligatorias:

- id (UUID)
- document (VARCHAR)
- candidate_id (INTEGER)
- mesa_id (INTEGER)
- timestamp (BIGINT)

Al cerrar la jornada electoral, lee de la tabla Votos la cantidad total de votos por candidato en todo el sistema y produce el archivo resume.csv.

Base de datos (PostgreSQL)

Tabla única: Votos.

Al término de la jornada, los datos aquí contenidos sirven como única fuente de la verdad para generación de CSV's definitivos y validación de unicidad.

Para resumir, el flujo del voto es el siguiente:

1. **Ingreso del voto en el Nodo de votación:** El ciudadano introduce su documento y selecciona su candidato a través de la interfaz CLI del Nodo.
2. **Envío del voto al Lugar de votación:** El Nodo construye el objeto Voto (incluyendo documento, candidato y mesa) y lo transmite localmente al proceso del Lugar de votación.
3. **Transmisión al Servicio de votación:** El Lugar de votación valida localmente que el documento no haya votado antes en esa estación y, mediante un mecanismo de reintentos (Reliable Messaging), envía el Voto al Servicio central por ICE.
4. **Persistencia en la Base de datos:** El Servicio de votación recibe el Voto, verifica que no exista duplicado en la tabla Votos y, de ser único, lo inserta en la base de datos.

Metodología de prueba

Entorno:

- Tres instancias de Lugar de votación (una por cada mesa: 1, 2 y 3). Cada una expone un servicio ICE local.
- Tres instancias de Nodo de votación (CLI), cada una asociada a su respectiva mesa y a uno de los Lugares.
- Una instancia de Servicio de votación (ICE central) conectada a la Base de datos PostgreSQL.
- Una instancia de base de datos PostgreSQL que contiene únicamente la tabla Votos.

No se probará la confiabilidad de la comunicación entre el nodo de votación y el sitio de votación. Caso contrario a la comunicación entre el sitio de votación y el servicio de votación. Se puede simular interrumpiendo el proceso del servicio de votación y volviéndolo a reanudar.

David Artunduaga, Rony Ordoñez, Juan de la Pava, Diego Polanco

Generación de datos:

Se crean 1000 votos de prueba, repartidos así:

- Mesa 1: 333 votos
- Mesa 2: 333 votos
- Mesa 3: 334 votos

Cada voto contiene:

- documento (cadena única generada automáticamente)
- candidate_id (entero entre 1 y 3)
- mesa_id (1, 2 o 3)
- timestamp (marca de tiempo en milisegundos)

Logs:

Nodo de votación (CLI):

- ENVÍA_VOTO: document candidateId mesaId
- ACK_RECIBIDO: document mesaId
  (cuando Lugar confirma recepción local)
- ERROR_ENVÍO: document motivo TIMEOUT o EXCEPCIÓN
  (en caso de fallo de comunicación con Lugar, si llegara a ocurrir)

Lugar de votación:

- RECHAZA_DUPLICADO_LOCAL: document
  (si la cédula ya figura en su conjunto local)
- ENVÍA_RM: document intento=N
  (cada vez que intenta enviar el voto al Servicio)
- REINTENTO_RM: document intento=N, causa=TIMEOUT o EXCEPCIÓN
- ACK_SERVICIO: document intento=N
  (cuando el Servicio confirma que persistió o rechazó por duplicado)
- VOTO_NO_ENVIADO: document tras N intentos
  (si los reintentos fallan)

Servicio de votación:

- RECIBE_VOTO: document candidateId mesaId timestamp
- PERSISTE_VOTO: document candidateId mesaId
- VOTO_DUPLICADO: document mesaId
  (al detectar índice único)

- ERROR_PERSISTENCIA: document causa
  (si falla la inserción)

Al cierre de jornada (para la entrega final):

- START_CLOSE_ELECTION
- WRITE_RESUME_CSV
- WRITE_PARTIAL_CSV: mesaId
- END_CLOSE_ELECTION

Consultas SQL para verificar:

- Conteo total de votos persistidos:

SELECT COUNT(*) AS total_votos FROM Votos;

- Conteo de votos únicos por documento y mesa

SELECT COUNT(DISTINCT document, mesa_id) AS votos_unicos, COUNT(*) AS total_votos FROM Votos;

- Verificación de duplicados (no debe devolver filas)

SELECT document, mesa_id, COUNT(*) AS ocurrencias FROM Votos GROUP BY document, mesa_id HAVING COUNT(*) > 1;

Escenarios de prueba

Escenario 1, Flujo ideal:

Objetivo: Verificar que, cuando el Servicio está disponible en todo momento, los 1000 votos se persisten correctamente sin duplicados ni reintentos.

Configuración:

- El Servicio de votación corre sin interrupción.
- Los tres Lugares (Mesa 1, Mesa 2, Mesa 3) están activos y en línea.
- Cada Nodo de votación contiene un lote de votos:
  - Mesa 1: 333 votos
  - Mesa 2: 333 votos
  - Mesa 3: 334 votos

Pasos a ejecutar:

1. Iniciar base de datos

2. Iniciar los lugares de votación
3. Iniciar simultáneamente los 3 nodos y sus respectivos votos
4. Esperar 20 segundos para detener los procesos
5. Corroborar estado de la base de datos

Estado final:

- Nodos (sumados los 3):
- ENVÍA_VOTO total = 1 000
- ACK_RECIBIDO total = 1 000
- ERROR_ENVÍO = 0

Lugares (sumados los 3):

- ENVÍA_RM total = 1 000 (333 + 333 + 334)
- REINTENTO_RM total = 0
- ACK_SERVICIO total = 1 000

Servicio de votación:

- RECIBE_VOTO = 1 000
- PERSISTE_VOTO = 1 000
- VOTO_DUPLICADO = 0

Base de datos:

- Filas totales en Votos = 1 000
- Filas únicas (document, mesa_id) = 1 000
- Duplicados = 0 filas

Escenario 2, Fallo temporal del servicio:

Objetivo: Confirmar que, si el Servicio de votación se interrumpe temporalmente, los Lugares reintentan y, al restablecerse el Servicio, no se pierde ningún voto.

Configuración:

- Mismas condiciones que en el Escenario 1 (Servicio y Lugares activos).
- Cuando el Servicio haya procesado aproximadamente **100 votos**, se detendrá el proceso del Servicio durante **5 segundos**.
- Durante esos 5 segundos:

    - Los Nodos siguen enviando votos al Lugar.
    - Cada Lugar intenta enviar al Servicio y registra reintentos.
- Pasados los 5 segundos, se reinicia el proceso del Servicio.

Pasos a ejecutar:

1. Iniciar Base de datos y Servicio de votación.
2. Arrancar los tres Lugares.
3. Iniciar los tres Nodos para que envíen sus 1 000 votos (50 ms entre cada uno).
4. Monitorear el log del Servicio; cuando PERSISTE_VOTO alcance **100**, detener el proceso del Servicio.
5. Durante los siguientes **5 segundos**, los Lugares registran reintentos (TIMEOUT_SERVICIO y REINTENTO_RM).
6. A los 5 segundos, reiniciar el Servicio.
7. Esperar 20 segundos adicionales para que los Lugares completen los envíos pendientes.
8. Detener los tres Lugares, los tres Nodos y finalmente el Servicio.
9. Corroborar el estado de la Base de datos.

Estado final esperado:

- **Lugares (cada uno):**
    - ENVÍA_RM total > 333/334 (incluye reintentos durante la caída).
    - REINTENTO_RM total = cantidad acumulada en esos 5 s.
    - ACK_SERVICIO final (sumado) = 1 000
- **Servicio de votación:**
    - PERSISTE_VOTO = 1 000
    - VOTO_DUPLICADO = 0
- **Base de datos:**
    - Filas totales = 1 000
    - Filas únicas = 1 000

    - Duplicados = 0 filas

- **Latencias (opcional):**
    - Votos procesados antes de la caída: latencia "envío Nodo → persistencia BD" < 100 ms.
    - Votos enviados durante la caída: latencia ≥ 5 s.

Escenario 3, Duplicado de votos:

Objetivo: Verificar que, si un Nodo de votación envía a propósito el mismo (document, mesa_id) dos veces, el Servicio persista el voto solo una vez.

Configuración:

- Solo participa la Mesa 1.
- Nodo 1 lee un archivo con 100 votos organizado como sigue:
    1. Primeros 50 votos con documentos únicos.
    2. Reenvío de 20 de esos 50 (duplicados, sin intervalo).
    3. Envío de los 50 votos restantes con intervalo de
- El Servicio permanece activo sin interrupciones.

Pasos a ejecutar:

1. Iniciar Base de datos y Servicio de votación.
2. Arrancar Lugar 1.
3. Ejecutar Nodo 1 modificado para duplicar 20 votos después de recibir cada ACK.
4. Esperar 5 segundos tras completar los 100 envíos (50 originales + 20 duplicados + 50 finales).
5. Detener Lugar 1 y, luego, el Servicio.
6. Corroborar el estado de la Base de datos.

Estado final esperado:

- **Nodo 1:**

    - ENVÍA_VOTO total = 120 (50 originales + 20 duplicados + 50 finales)
- **Lugar 1:**
    - ENVÍA_RM total = 120
    - REINTENTO_RM = 0 (no hubo fallo de conexión)
    - ACK_SERVICIO = 120
- **Servicio de votación:**
    - RECIBE_VOTO = 120
    - PERSISTE_VOTO = 100
    - VOTO_DUPLICADO = 20
- **Base de datos:**
    - Filas totales = 100
    - Filas únicas = 100
    - Duplicados = 0 filas

Escenario 4, Caída y recuperación del servicio:

Objetivo: Comprobar que, si el Servicio de votación se detiene abruptamente mientras los Lugares siguen recibiendo votos, al reanudar el Servicio los votos pendientes se persisten sin pérdida ni duplicados.

Objetivo: Comprobar que, si el Servicio de votación se detiene abruptamente mientras los Lugares siguen recibiendo votos, al reanudar el Servicio los votos pendientes se persisten sin pérdida ni duplicados.

Configuración:

- Tres mesas (Lugar 1, Lugar 2, Lugar 3) planean enviar **1 000 votos** (333 + 333 + 334).
- El Servicio procesa los primeros **200 votos** y luego se detiene.
- El Servicio permanece apagado durante **10 segundos**.
- Durante esos 10 segundos, los Lugares siguen recibiendo votos de los Nodos y registran reintentos cada 2 s (hasta 5 intentos por voto).

- Pasados los 10 segundos, se reinicia el Servicio en el mismo puerto y con la misma Base de datos.

Pasos a ejecutar:

1. Iniciar Base de datos y Servicio de votación.
2. Arrancar los tres Lugares.
3. Iniciar los tres Nodos para que envíen 1 000 votos con intervalo de **50 ms**.
4. Monitorear BD; cuando PERSISTE_VOTO alcance **200**, detener el proceso del Servicio.
5. Durante los siguientes **10 segundos**, los Lugares registran TIMEOUT_SERVICIO y REINTENTO_RM.
6. Tras 10 segundos, reiniciar el Servicio.
7. Los Lugares retoman los envíos pendientes y reciben ACK_SERVICIO hasta completar 1 000.
8. Esperar **5 segundos** tras el último ACK.
9. Detener los tres Lugares, los tres Nodos y finalmente el Servicio.
10. Corroborar el estado de la Base de datos.

Estado final esperado:

- **Lugares (cada mesa):**
    - ENVÍA_RM total = 1 000 + (reintentos durante caída)
    - REINTENTO_RM total = cantidad acumulada en esos 10 segundos
    - ACK_SERVICIO final = 1 000
- **Servicio de votación:**
    - RECIBE_VOTO total = 1 000
    - PERSISTE_VOTO total = 1 000
    - VOTO_DUPLICADO = 0
- **Base de datos:**
    - Filas totales = 1 000

- Filas únicas = 1 000
- Duplicados = 0 filas